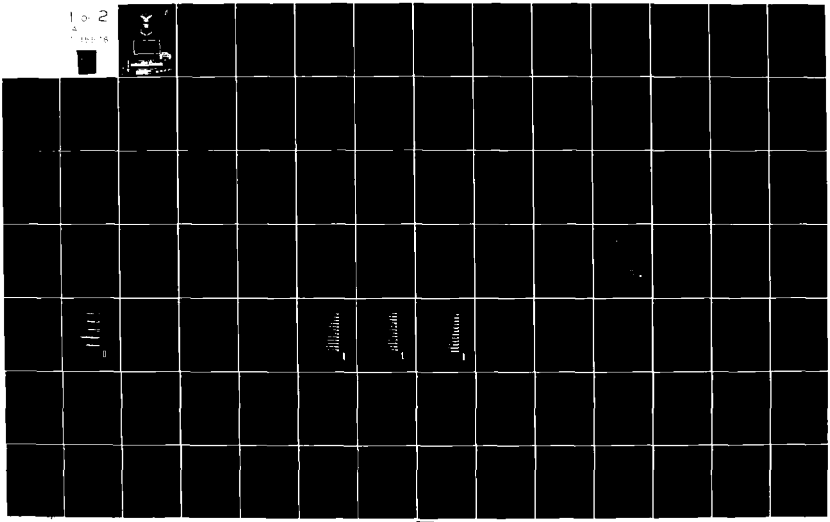
AD-A115 578 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
THRUPUT ANALYSIS OF AFLC CYBER 73 COMPUTERS.(U)
DEC 81 L T BONDURANT
UNCLASSIFIED AFIT/GCS/EE/81D-2 NL

1 of 2



AFIT/GCS/EE/81D-2

THRUPUT ANALYSIS
OF AFLC CYBER 73 COMPUTERS

THESIS

AFIT/GCS/EE/81D-2 Le Roy T. Bondurant
Civ USAF

AFIT/GCS/EE/81D-2

THRUPUT ANALYSIS
OF AFLC CYBER 73 COMPUTERS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air university

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Le Roy T. Bondurant

Civ USAF

Graduate Computer Science

December 1981

Accession For		
NTIS GRA&I		☒
DTIC TAB		☐
Unannounced		☐
Justification		
By		
Distribution/		
Availability Codes		
Dist	Avail and/or Special	
A		

PREFACE

This thesis is the result of a request from AFLC/LMT for assistance in conducting a Performance Analysis of their near capacity-filled CYBER 73 computers which were expecting additional workloads in the near future. The problem was investigated using exhaustive sampling and statistical techniques against typical workloads of actual daily, weekly and monthly computer cycles of accounting data.

The problem was vast and complex such that without the aid of prior studies by former AFIT students Dodson, Bear, Thompson, Blitt and Heidenreich, the effort may have been fruitless.

Many Datacenter personnel, including my sponsors, Maj Abbott and Lt. Morgan, Operations chief, Mr. Davis, and ASD Software analyst, Mr. Merchant, along with Librarian, Mrs. Kendall, who helped find volumes of background material, were generous with their assistance. Lt Col Bexfield and Maj Ross, as committee members, and especially Dr Hartrum, my advisor, provided valuable guidance and direction during the study.

Special appreciation is due my wife, Vivian, for not only her daily encouragement and support but also for her vital assistance in proofreading and editing this document.

LeRoy T. Bondurant

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Supporting a request from AFLC/LMTA for assistance in evaluating the performance of their CYBER-73 computer system on an expanding workload, this thesis models and analyzes the existing workload in an effort to explain job elongation in turnaround time and to predict future performance under hypothesized changes. The workload of a selected typical month, July, 1981, was used as a baseline for the study. Through a systematic reduction and parameterization of job characteristics, the baseline data was clustered and statistically analyzed to detect trends in areas where job delay could occur. Performance models of elongation areas were subsequently developed, tuned, and varied to permit maximum workload optimization for the computer system. A 31 percent reduction in turnaround time was ultimately predicted from a series of 6 recommended changes in current procedures leading to a potential annual savings in excess of two and a third million dollars for the command's six computer installations.

THRUPUT ANALYSIS

OF AFLC CYBER 73 COMPUTERS

I. INTRODUCTION

Any data processing function is responsible for the delivery of computer services to its users. This responsibility includes the efficient and effective use of datacenter resources - hardware, software support systems, operational schedule, and the installation rules and procedures controlling the production process. That utilization, together with the factors that effect utilization (e.g., hardware/software failures), establishes the service levels provided to the user. These levels may be expressed in terms of the timeliness, accuracy, cost, and reliability of datacenter processing.

The prime objective of data processing management should be to hold these levels within certain limits for the current workload and to plan for future conditions which may cause increases in the workload such that, where practical, these service levels are maintained or improved.

THE CPE FUNCTION

Computer Performance Evaluation (CPE) techniques are used to assist the DP manager in meeting those objectives. Through CPE, management usually undertakes one or more of the management tasks of forecasting, resource planning and pricing, operational analysis, and performance reporting (Ref: 35). Al require that basic

performance information be available if the tasks are to be performed effectively. These data are customarily made available through data collection activities which operate in the data processing environment. The degree to which the collected data are tailored, in part, determines the effectiveness with which the management objectives are met. Thus a CPE analyst, together with a description of a system's hardware, software, and current and projected workloads, can provide management all the information necessary to predict the system's thruput rate, response times, resource utilization, and the point at which resources will be exhausted (Ref: 26).

BACKGROUND

Since January, 1978, AFLC has had an ongoing effort to convert computer application systems from their aging IBM 7080 computers to the CDC CYBER 73 computers at headquarters and its five Air Logistics Centers (ALCs) (Ref 2: Attachment 1, 2). Early simulations and sizing studies estimated thruput ratios from 1.5 to 7:1 of the CYBER vs the 7080 was possible (Ref 2: Attachment 1, 2). Actual conversions however, achieved only a 3:1 ratio (Ref 2:14). Thus a 3:1 ratio, although considered conservative, was accepted as the performance guideline for the remaining conversions (Ref 2:14). This decision permitted a fast conversion effort with minimum programmer/analyst experience (Ref 34).

Recently, as the conversion effort nears completion, a problem has arisen in getting computer products to the user within acceptable time frames. While CPU times have been remarkably close to the 3:1 ratio, the turnaround times (time from job input until output of the final line of print) are increasing with each additional system converted (Ref 1:2).

Moreover, many of the large data-file and machine-time-consuming systems were not included in the earlier conversion efforts and are waiting for conversion when time permits. Given the existing scheduled workload, doubt exists as to whether the current CYBER hardware configuration can provide the required mission support when those systems are finally converted and added to the job mix (Ref 1:2).

PROBLEM STATEMENT

AFLC's conversion of application systems from the IBM 7080 to the CDC CYBER 73 computers is not achieving the performance estimate specified in sizing studies accomplished prior to the conversion. Additionally, statistics on some converted systems indicate a gradual degradation of performance as more systems are converted and added to the job mix.

AFLC/LMT requested a thruput analysis of the CYBER 73 computer to determine if and where a bottleneck existed, and what measures should be taken to alleviate the problem.

SCOPE

This thesis analyzes the performance and recommends solutions for the workload of the headquarters AFLC CYBER 73 only. While not specifically investigating similar characteristics of the ALC computers, it is anticipated that the recommendations for problem solution will apply to both headquarters and ALC CYBER 73 systems.

A preliminary resource analysis accomplished by LMT personnel revealed that during certain periods i.e., 0000-0800, the machine is normally reserved for the large

resource-consuming programs. These programs have been determined to be both I/O and CPU bound. This leaves little room for system thruput improvement unless improvements are also made in the user programs as well (e.g., efficiency in coding, blocking, etc.) (Ref 20:332-335). It was also ascertained that the job entry queue (those jobs waiting for initial resource allocation) was substantially overloaded which negated the effects of many quick-fix schedule-tuning methods (Ref 31: Interviews). Hence, much of the analysis effort of this thesis centers around an analysis of the computer's internal activity. The initial LMT findings, however, saved considerable time in developing hypotheses of the problem areas.

The developed hypotheses and analysis methods are accomplished completely through Computer Performance Evaluation (CPE) techniques advanced and advocated by the IEEE (Ref: 33) and the AFIT School of Engineering (Refs 21:28).

APPROACH

The approach used to analyze and resolve the thruput problem includes:

(1) Understand the system

(2) Characterize the workload

(3) Develop data reduction/extraction programs to be used to reduce DAYFILE data tapes to manageable workload parameters

(4) Select statistical programs for use in workload data analysis

(5) Select a representative mix of the workload to serve as a benchmark for testing

(6) Hypothesize possible bottleneck areas

(7) Develop an Analytic Model characteristic of the system to use for testing the hypotheses

(8) Drive the model under varying characteristic changes to determine what effect different configuration combinations have on a static workload

(9) Develop conclusions on which to base recommendations

(10) Use the model to verify hypotheses and recommendations

(11) Implement the most promising changes within the range that AFLC will allow

II. METHODOLOGY

TOOLS AND TECHNIQUES

FORWARD

This chapter discusses the methods and procedures used to accomplish the goals of the thesis. Many of the methods discussed are typical for computer performance techniques employed as state-of-the-art procedures in evaluating computers for thruput performance. Although some aspects of the system under investigation were known beforehand to provide poor performance (e.g., input queue overloading, priority by resource use, etc.) they were nevertheless used along with developed hypotheses to theorize the overall problem (Ref 31: Interviews).

UNDERSTAND THE SYSTEM

One of the initial tasks in performing the evaluation of a computer system involves understanding the particular computer system. Five categories of information were gathered from AFLC personnel directly responsible for the day to day operation of the CYBER 73 concerning (1) management organization of the installation, (2) characteristics of the workloads processed by the computer, (3) descriptions of the hardware configuration, (4) software programs in use, and (5) information as to what computer-usage data are collected (Ref 5).

WORKLOAD CHARACTERIZATION

The workload of a computer is defined as the set of all inputs (programs, data, commands) the system receives from its environment (Ref 20:221). For the purpose of a thruput analysis of the system, software programs, such as compilers, operating system algorithms, deadstarts, and scheduling, are considered to be part of the system overhead and not included with the job steps defined as the workload (Refs 3:19; 4:1). Accordingly, the bounds of the workload for this thesis consists of the specific computer programs which must be processed by the computer in order to satisfy user requests. These steps are further delineated by hourly, daily, weekly, and monthly periods. An additional category, lifetime, extrapolates expected workload increases for the purpose of hypothesizing when resources will be exhausted.

One widely used method for characterizing workloads on computer systems includes modeling jobs by resource usage and event time techniques such as a Poisson distribution process to mathematically represent the arrival, execution, and departure of job steps characteristic of a system's real workload (Ref 19:45). This technique offers stability, reproducibility, flexibility, and brevity to the model since the actual workloads of multiprogrammed computers are extremely variable and generally not reproducible (Ref 4:4).

In fact, the most accurate performance values will be obtained when the system is measured under its actual workload. The CYBER, through its use of an event driven software monitor in the form of a DAYFILE of all user job steps and activities (Ref 13: Chapter 2-7), provides an accurate portrayal of the system's actual workload (Ref: 11). Moreover, The workload of the CYBER-73 is stable although escalating over the long term, with the same production jobs run week after week. Thus, the workload model for this thesis consists of a sampling of job step extracts from the computer's DAYFILE tapes. Table 1 shows the features about the job used to describe the workload.

TABLE I

VARIABLES AND WORKLOAD CHARACTERIZATION

VARIABLE	DESCRIPTION
JOBNAME	NAME & organization submitting job
DAYRUN	day of week job was run
ARRIVAL	time of arrival at control poing
CPUUSAGE	CP time for execution
MEMUSED	max core used
IOUSE	total I/O time during execution
CMTIME	Central Memory usage in word-min
PPTIME	total Peripheral Processor time
SS	system seconds used
DISKIO	disk requests during execution
TAPEIO	tape I/O requests during execution
INQTIME	time waiting in resource queues
7TRACK	7 track tape drives used
9TRACK	9 track drives used
7TRACKREQ	7 track drives requested
9TRACKREQ	9 track drives requested
TIMEREQ	job card time requested
CMREQ	job card CM requested
TURNAROUND	time required to complete job
TAPEWAIT	time waiting for tape mounting
POSTPROC	time spent printing
SORTTIME	time from start sort to end of sort
DISKWAIT	time waiting access to disk
LINES	lines printed

REDUCTION PROGRAMS

A package of three programs were developed to reduce, sort, extract, and parameterize the DAYFILE data (see Appendix A).

(REDUCTN) This is a PASCAL reduction algorithm which reads through weekly or daily DAYFILE tapes to extract jobstep statistics pertaining to user resource usage. Events excluded from the extract are activities relating to system overhead, user generated messages, resource use clarification (e.g., catalog), and recovered error conditions. Also excluded are jobs which started prior to or are still running at the end of the recorded day (0000 - 2400). Table 2 is an example of a partial DAYFILE with key data used in the REDUCTN program.

(SORTDF) This is a Sort routine to sort the output of the extraction run by jobname and time.

(PARAMEX) This is a PASCAL algorithm which receives the output from SORTDF and builds the workload parameter file. Job step events are systematically read and parameters extracted (or derived as in the case of TURNAROUND time, DISK/TAPE waiting, etc.). These are subsequently written to a statistical parameter file for input to SPSS runs for analysis. Table 3 portrays the input format and location of parameter fields for PARAMEX. Table 4 shows the output parameters and length.

TABLE 2

DAYFILE EXAMPLE WITH KEY DATA

TIME	NAME	DATA FIELD
23.57.32.	MOP2F47.	** TOTAL RECORDS SORTED 000059680
23.57.32.	MOP2F46.	CATLOG,MI022X0 MI022F0
00.13.33.	MOP2F47.	LABEL WRITTEN WAS NIOF22FI
00.13.33.	MOP2F47.	NT66 BLOCKS WRITTEN - 008965
00.13.33.	MOP2F47.	NT66 BLOCKS READ 008965
00.24.13.	MOP2F47.	RETURN, MIO22F0..
00.24.14.	MOP2F47.$OP	00001344 WORDS - FILE OUTPUT, DC 40
00.25.51.	MOP2F47.$MS	3584 WORDS (16773129 MAX USED)
00.25.51.	MOP2F47.$CPA	713.650 SEC. 713.650 ADJ
00.25.51.	MOP2F47.$IO	51108.927 KWS. 1547.382 ADJ
00.25.51.	MOP2F47.$CM	52806.451 KWS. 3119.441 ADJ
00.25.51.	MOP2F47.$SS	5380.474
00.25.51.	MOP2F47.$ACCESSES	43108
00.25.51.	MOP2F47.$PP	2325.437 SEC. DATE 06/21/81
00.25.59.	MOP2F47.	0000424 LINES PRINTED,LQ12
00.36.14.	MOP2H48.	UNLOCK
00.48.42.	MOP2H48.	(MT 076 ASSIGNED)
00.48.43.	MOP2H48.	MT76 BLOCKS WRITTEN - 006347
02.30.42.	MOPM059.	0000006 CARDS READ, CR10
02.30.42.	MOPM059.	ENTERED INPUT QUEUE
02.30.45.	MOPMO59.$AC	PR00010 TL000500 I0000000 CM060000
02.30.45.	MOPMO59.$AC	MT000001 NT000000
02.30.45.	MOPMO59.$IP	00000192 WORDS - FILE INPUT , DC 00
02.30.47.	MOPMO59	MOPMO,MT1,DM000,T500,AC=MOMO.
02.30.48.	MOPMO59.	MOUNT,SN=TULIB01,VSN-TULIBR.
02.30.49.	MOPMO59.	MOUNTED VSN=TULIB01,SN-TULIB01,EST=73
02.30.50.	MOPMO59.	ATTACH,MOUCL,ID=MOVPM,SN=TULIB01,EST=00
02.30.56.	MOPMO59.	ALREADY MOUNTED VSN=TULIBR,SN=TULIB01,EST=00
02.30.59.	MOP2H48.	SORTMRG,I=PROC,I=SORTOUT
02.40.21.	MOPMO59.	REWIND,ZZBODLI.

TABLE 3

INPUT DATA FORMAT FOR DAYFILE EXTRACT

KEY	KEY LOC	DATA ELEMENT EXTRACTED	DATA LOC
$MS	20-22	MASS STORAGE USED	25-30
$CPA	20-23	CPU TIME	26-33
$IO	20-22	I/O TIME	43-50
$CM	20-22	CM TIME	43-50
$SS	20-22	TOTAL EXECUTION TIME	43-51
$ACC	20-23	DISK ACCESSES	33-38
$AC	20-22	7 TRACK REQUESTED	30-31
$AC	20-22	9 TRACK REQUESTED	40-41
$AC	20-22	TIME REQUESTED	37-40
$AC	20-22	CM REQUESTED	55-60
$PP	20-22	PERIPHERAL PROCESSOR TIME	25-34
$PP	20-22	DAY OF RUN	52-53
$EJ	20-22	JOB COMPLETION	2-10
LINES	29-32	LINES PRINTED	21-27
MT	23-24	7 TRACK TAPE ASSIGNED	27-28
NT	23-24	9 TRACK TAPE ASSIGNED	27-28
BLOCKS	27-32	TAPE I/Os	43-48
MOUNTED	21-27	DISK ASSIGNMENT	2-10
MOUNT,	21-26	DISK DRIVE REQ	56-57
ALREADY	21-27	DISK REQ ACKNOWLEDGE	2-10
SORTM	21-25	START SORT	2-10
TOTAL	25-29	RECORDS SORTED/END SORT	2-10
REQUEST	29-38	REQUEST FOR TAPE	2-10
ENTERED	21-27	JOB ENTERED INPUT QUEUE	2-10

TABLE 4

OUTPUT PARAMETERS OF DAYFILE EXTRACT

PARAMETER	FIELD
NAME	XXXXXX
DAY	00
ARRIVAL	00.0000
CPU	0000
MEM	000000
IO	0000
CMTIME	0000
PPTIME	0000
SS	0000
DISKIO	000
TAPEIO	000
INQTIME	00.0000
7TRACK	00
9TRACK	00
7TRACKREQ	0
9TRACKREQ	0
TIMEREQ	0000
CMREQ	000000
TURNAROUND	00.0000
WAITTIME	0000
POSTPROC	0000
SORTTIME	0000
DISKWAIT	0000
LINES	000000

STATISTICAL PROGRAMS

The Statistical Package for the Social Sciences (SPSS) (Ref: 34) is an integrated system of computer programs designed for the analysis of a wide variety of accounting data to simplify the process of data analysis in a convenient manner. This thesis employs SPSS almost exclusively for the analysis and hypothesis testing of the workload parameters used to theorize the thruput problem (see Chapter V Hypotheses). Runs for levels of measurement on nominal, ordinal, interval, and ratio properties are employed using CONDESCRIPTIVE, FREQUENCIES T-Test, CROSSTABS, Bivariate Correlation, Multiple Regression, and Factor analysis techniques.

CONDESCRIPTIVE runs provide measures of central tendency and dispersions about the mean (Ref 34:18). Such tendencies as average I/O time required for jobs, mean number of disk or tape accesses, overall resource-wait, etc., can be directly provided by CONDESCRIPTIVE, to permit the quick development of an initial hypothesis prior to exhaustive testing.

FREQUENCIES computes and presents one-way frequency distribution tables of discrete or categorical variables (Ref 34:194). Clusters of similarly grouped data may be selected when kurtosis is suspected or when the range of values is too great. As with CONDESCRIPTIVE, this procedure is also conducive to initial hypothesis development leading to a theory on where problems are occurring.

T-TEST provides the capability of computing probability levels for testing whether or not the difference between two sample means is significant (Ref 34:267). This test permits the evaluation and detection of differences between effects rather than the effects themselves. With such a test, measured sample periods in which more or less

jobs are processed, or in which larger or smaller jobs are processed, may be effectively evaluated (e.g., Saturday vs Tuesday processing). Thus, since it is highly probable that the two samples would be different due to natural variability in the population, a difference in sample means may be tested for significance (whether there exists a true difference between the two populations) using the T-TEST.

CROSSTABS provides an easy to use joint frequency distribution of variable cases according to two or more classifications of variables. Using CROSSTABS a quick perusal of its output tables gives indications of how dramatic a difference is between sets of data (Ref 34:220). Thus, hypothesis tests, based on statistical results from other SPSS routines, e.g., FREQUENCIES, can be confirmed or denyed through the use of CROSSTABS procedures. On the other hand, if all reasonable test variables fail to reject a hypothesis, while not having proved the contention, at least efforts to falsify it proved to be fruitless (Ref 34:222).

BIVARIATE CORRELATION provides a single number which summarizes the relationship between two variables, such as I/O usage vis-a-vis TURNAROUND. The correlation indicates the degree to which variation in one variable is related to variations in the other (Ref: 23). The procedure may be expanded to provide a means for comparing the strength of relationships between one pair of variables and a different pair. The method, nonparametric, which assumes randomly distributed cases of variables, is less stringent and detailed than the CROSSTABS procedure yet provides a quick and easy analysis of variables which may be used to supplement CROSSTABS. The method is particularly useful with a large number of categories or ranks of variables and requires nothing more than an ordinal level of measurement.

MULTIPLE REGRESSION uses generalized techniques of coorelation analysis providing measurements of data on a large number of independent variables (Ref: 22). Multiple Regression techniques are used as descriptive tools to find the "best" predictive equation and evaluate its predictive accuracy and as a control for other factors in order to evaluate the contribution of a specific variable or variables on a theorized phenomenon.

Hence, Multiple Regression provides a good tool for explaining system performance. As an example, a regression of TURNAROUND time against CPU Usage producing a Correlation Coefficient of .98, would indicate that jobs were spending almost all of their time in the CPU while other factors, I/O, etc., had little impact on TURNAROUND (Ref: 23). Thus regression techniques, particularly a stepwise regression where a dependent variable (possibly TURNAROUND time) is successively regressed against independent variables, until the group explaining most of the variation in the dependent variable emerges, will be used for most primary hypotheses development.

FACTOR ANALYSIS, an iterative correlation procedure, provides a data reduction capability for ease in rearrangement, or reduction of data to a smaller set of factors or components that may be taken as source variables for the observed interrelations in the data (Ref 34:469). More common uses of the procedure include exploratory - the exploration and detection of patterning of variables with a view of the discovery of new concepts and a possible reduction of data; confirmatory uses - testing of hypotheses about the structuring of variables in terms of the expected number of significant factors and factor loadings; and uses as a measuring device - the construction of indicies to be used as new variables in later analysis.

The method applied in this thesis will primarily consist of confirmatory, or hypothesis testing. That is, to test a premise that there may exist some underlying regularity in the observed correlation of a variable which is influenced by various determinants, some of which are shared by other variables in the set.

BENCHMARKING

Benchmark is defined by Hatt as a set of programs characteristic of a user's or a system's workload which is provided to commercial manufacturers for the purpose of testing the performance of a computer under operational conditions - whether it can do the jobs and how long it takes (Ref 9:1). Futhermore, Goff (Ref 9:ix) suggests that for a computer evaluation, a benchmark should be comprised of:

(1) a mix of jobs that is representative of the user's projected workload over the life of the system

(2) Demonstrations of data storage equipment and techniques.

(3) Computer programs designed to test specific functions.

For the purpose of this thesis, benchmarking is limited to (1) above - establishing a workload representative of the present and projected workload of the CYBER-73. This workload will be derived, measured, and statistically evaluated to assure consistency with the system's real workload and thereafter called the benchmark. Subsequently, models characterizing the functions and resources of the system will be developed and driven by the benchmark to evaluate performance under existing and projected loads, and under varying configuration mixes for explanatory and predictive purposes.

MODELING

One of the more useful and practical tools for explaining and predicting performance on computer systems is the technique of modeling. Buzen defines modeling as a computer program that receives, as input, a description of a system's hardware, software and workload which on the basis of the given data, calculates what the performance of the system will be (Ref 12:2). Modeling techniques used for this performance evaluation include the analytic techniques of queueing theory and multivariate analysis (Ref: 23)

ANALYTIC MODELS are essentially a set of equations that describe the performance of the system being studied. These models, used chiefly for predictive purposes, are called upon to evaluate the effects of various changes to a computer system's internal components and external workloads. Analytic models provide a means for assessing their impact before any changes are actually made (Ref 12:3).

QUEUEING models provide the basis for most analytic modeling. Queueing models are characterized by waiting lines, or queues, leading to some server. The servers correspond to the components of the computer - I/O devices, CPUs, channels. Although much simpler to develop than simulators. A queueing model must make many assumptions about distributions which cannot be met in many complex systems (Ref 22:65).

MULTIVARIATE analysis techniques for modeling have been used sparingly in the evaluation of the performance of computer systems and, generally, because of the complexities of most large systems, problems of linearity, additivity, and non-multicollinearity tend to complicate the interpretation of the results of this

approach, raising questions about the interpretive conclusions reached. However, many of the problems inherent with simulation and queueing models, such as the inability to model software and scheduling algorithms, are not present to degrade the predictive capability of multivariate analysis (Ref 23:65).

Through multilinear regression, factor analysis, and discriminant analysis, actual accounting data from a system's typical workload can be analyzed to develop hypotheses about the performance of a similar workload given a change in the characteristics of one or more of the system's components. Subsequent analysis of variants (after the actual or simulated change) would provide sufficient verification that the hypotheses did, indeed, prove correct (Ref 23:67).

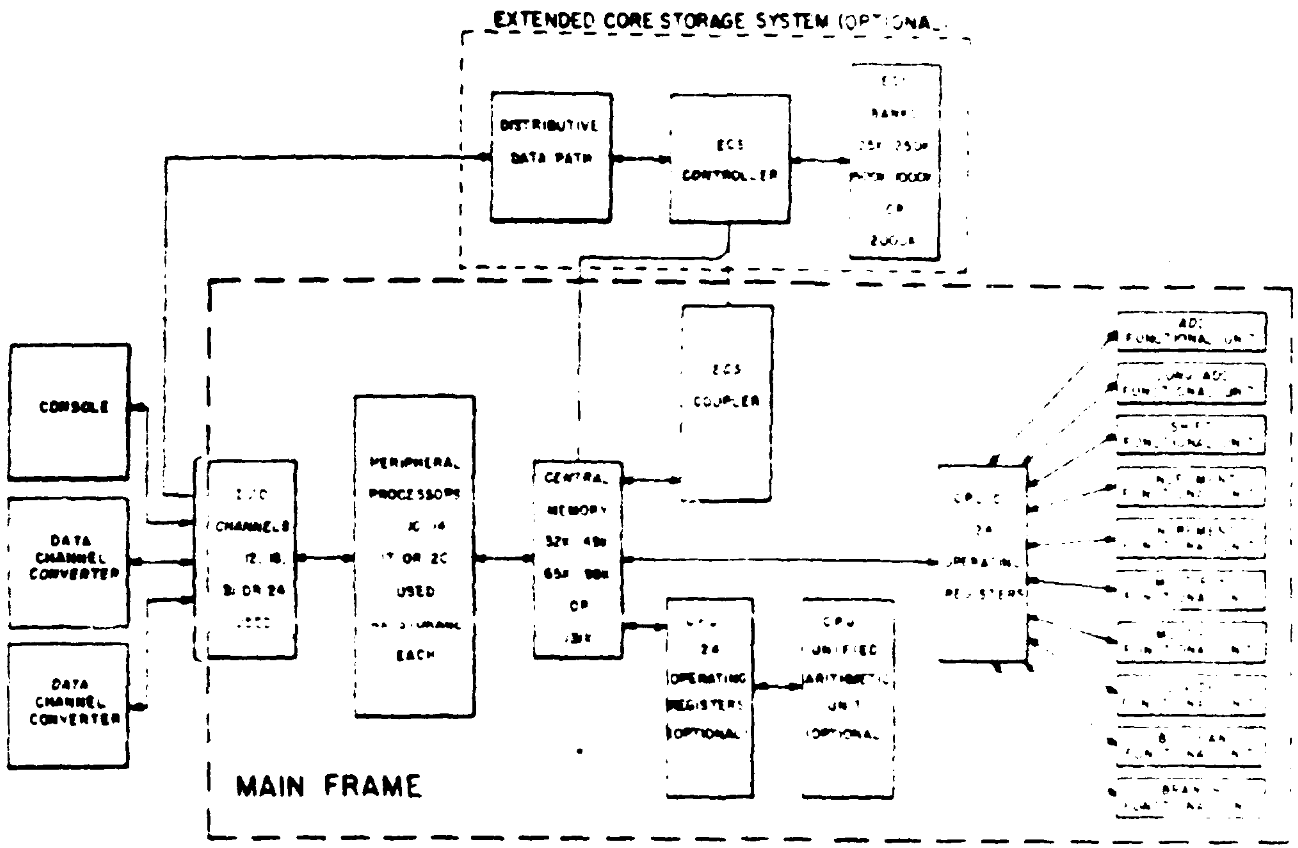


Fig 1. CYBER Mainframe Configuration (Ref 15)

III THE AFLC CYBER 73 ENVIRONMENT

INTRODUCTION

This chapter describes the salient features of the CDC CYBER 73 system and its environment organization as configured at Headquarters, AFLC. The purpose here is to provide a description of those system characteristics which may serve as a background for the workload and performance analysis carried out in later chapters and not to delve into lengthy details of the full system. The computer itself can consist of additional features, not all of which pertain to the system as installed at AFLC, and not all of which are pertinent to the problem analysis. The Control Data Corporation's 6000 series documentation, references 14 through 17, explains in detail the specific characteristics of a fully configured CYBER 73 system and its capabilities. Extracted references to hardware and software features are primarily from those manuals.

SYSTEM OVERVIEW

The CYBER 73 is a multi-programmed, multi-processing, file oriented system consisting of a mainframe and a flexible assortment of peripheral and control equipment. The system is designed to overlap as many processes as possible through the use of independent processors called Peripheral Processors. The mainframe contains 20 of these Peripheral Processors and the data channels necessary to communicate with the peripheral equipment; a Central Memory (CM); a central processor unit (CPU), with 24 operating registers in the arithmetic unit; and the attendant control logic to drive the components (see Figure 1).

HARDWARE CONFIGURATION

CENTRAL PROCESSOR. The CPU is an arithmetic processor which communicates only with Central Memory. It is isolated from the Peripheral Processors and thus is free to carry on computations unencumbered by input/output requirements. Memory transfer rate is up to one word every 100 nanoseconds.

PERIPHERAL PROCESSORS are identical, small general purpose computers which operate independently of and simultaneously with the CPU and each other. Many programs thus may be running at the same time, or a combination of processors can be involved in one program requiring a variety of input/output tasks as well as the use of the Central Memory and the central processor. The CYBER 73 has two banks of ten peripheral processors tied to two sets of bi-directional paths to central memory. Up to four processors may be writing in Central Memory while another four are simultaneously reading from Central Memory (see Figure 2).

The CYBER 73 is a full multiprocessing system when dual CPU's are a part of the configuration. AFLC's system, however, is configured with just one CPU. Quasi multiprocessing is nonetheless accomplished with the perpherial processors acting as system control computers performing the slow input/output and supervisory operations while the Central Processor accomplishes the computational operations. Each Peripheral Processor has a 12 bit, 4096 word random-access memory with a cycle time of 1000 ns which, as a group of 10 over the bi-directional path, coincides with the minor cycle (100 ns) of the CPU. The cycle time of the PPs however, is not dependent on the timing of the CPU.

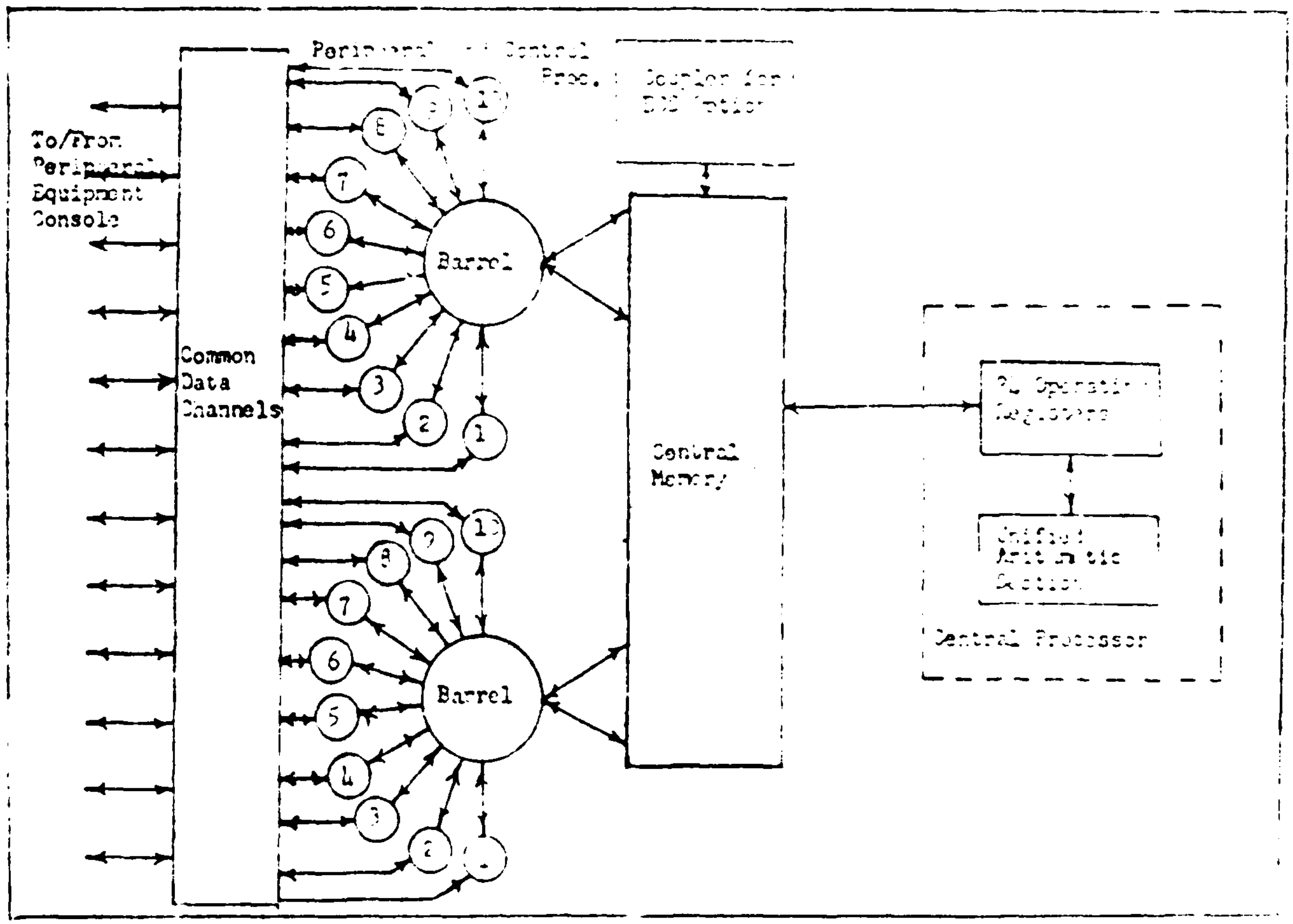


Fig 2. General Block Diagram of a CYBER 73 (Ref 15)

CHANNELS. Twenty-seven channels (24 I/O plus 3 utility) serve the components of the CYBER 73. The 20 Peripheral Processors communicate with external equipment via thirteen 12 bit (plus control) channels. Each channel (except 0 which is connected to the real time clock) connects to one or more devices. Although only one external device can utilize a channel at one time, all channels, like all PPs can be active simultaneously. Each channel operates at a maximum rate of one 12 bit word per microsecond. In addition, a special 128 bit interlock register tied to two access channels, each accomodating a bank of 10 Peripheral Processors, provides a means for the Peripheral Processors to communicate with each other without the necessity for making Central Memory references.

CENTRAL MEMORY is a core memory with a capacity of 131,000 60 bit words (1.3 million character positions) in 32 banks of 4096 words each. The banks are logically independent and may be phased into operation at 100 ns intervals. The Central Memory and data control mechanisms permit a word to move to or from Central Memory every 100 ns. Access to Central Memory from all areas of the system go to a common memory control and are issued to Central Memory via a bus line to all memory banks. The correct bank, if free, accepts the address and flags the memory control bus line to be cleared. The control accepts addresses from the various sources under a priority system at a maximum rate of one minor cycle and issues addresses to Central Memory at one every 100 ns. All 20 Peripheral Processors, in addition to the CPU have access to Central Memory.

PERIPHERAL DEVICES used with the CYBER 73 include thirty-three removable pack 844-2 disk drives, seven 659 800/1600 BPI nine track tape drives, nine 657 200/556 BPI seven track tape drives, two 405 card readers, three 512 line printers, and one 415 card punch unit. The 844-2 disk drives each have a capacity of 118 million 6 bit characters, an average access time of 40 ms, average rotational delay of 8.3 ms, with a

transfer rate of 1.13 million characters per second, providing a total on-line storage capacity of 3.9 billion characters. Two 7054 disk controllers permit simultaneous read and/or write operations within the disk subsystem. The 657 and 659 tape drives use half-inch, 2400 foot tape reels providing a max recording capacity of 23 and 37.4 million characters per reel, tape speed of 250 inches per second, and a recording density of 556 and 1600 Bits Per Inch (BPI), respectively. Table 5 is the fully configured device-channel relationship for the CYBER 73 as configured at Headquarters AFLC.

SOFTWARE CONFIGURATION

OPERATING SYSTEM. The operating system for the CDC CYBER 73 provides Supervisory Control Of Program Execution (SCOPE) and is thus in complete control of the computer. SCOPE accepts input in the form of jobs submitted by users and processes them as directed by control cards accompanying each job as well as by keyboard commands input by the operator.

TABLE 5

CYBER 73 DEVICE/CHANNEL RELATIONSHIP

NO.	DEVICE	CHANNEL(S)	CAPACITY
(1)	844 SYSTEM DISK	32	118m char
(1)	405 CARD READER	11	1200 CPM
(1)	405 CARD READER	04	1200 CPM
(2)	512 LINE PRINTER	04	1200 LPM
(1)	512 LINE PRINTER	11	1200 LPM
(1)	415 CARD PUNCH	11	250 CPM
(3)	844 PERM FILE DISK	24 & 27	118m char
(3)	844 DEVICE SET	24 & 27	118m char
(2)	844 PUBLIC DISK	24 & 27	118m char
(3)	844 PERM FILE DISK	25 & 30	118m char
(2)	844 PUBLIC DISK	25 & 30	118m char
(3)	844 DEVICE SET	25 & 30	118m char
(9)	844 DEVICE SET	26	118m char
(9)	844 DEVICE SET	31	118m char
(9)	657 TAPE UNIT	12 & 13	556 BPI
(8)	659 TAPE UNIT	22 & 23	1600 BPI
(1)	657 TAPE UNIT	22 & 23	556 BPI
(1)	2551 INTERCOM HOST	07	
(1)	6612 CONSOLE	10	
(1)	6676 DATA SET CONTROLLER	21	

SCOPE uses the Peripheral Processors (PP) for system input/output tasks and the Central Processor (CPU) to execute user and system jobs. Central Memory (CM) contains user programs with system software utilities located at the upper and lower ends. SCOPE uses the CPU to perform tasks of a computational nature while routing all input/output functions to the PPs. In this manner, SCOPE, by employing the CPU almost exclusively for program computations, assemblies, and executions, is able to efficiently utilize the CPU (Ref 17: Chapter 1-1).

A PP can be assigned to control, input/output, job scheduling, control card interpreting, system house-keeping and other tasks as required. In general, operations which tend to require physical movement, interactive responses, waiting, etc., are functions assigned to the PPs and not to the high speed CPU. Tasks are assigned one at a time to each PP by the system monitor (JANUS). When an assigned task is completed, PP signals JANUS which in turn assigns another task to the PP. JANUS operates in PPO and is in complete supervisory control of the hardware system (Ref 17: Chapter 1-3).

SCOPE uses a progressive queueing scheme to accomplish multiprogramming. That is, three levels of wait queues, each successively smaller in the number of jobs held, contain a job until it reaches the CPU for execution. The input queue accepts all jobs entering the system and holds them until a jobs resources have been assigned and the job scheduled for execution. The CM queue holds jobs waiting for central memory and control point assignment (described below). These are roll/swapped jobs waiting for reassignment to a control point and/or central memory. Upward to 50 jobs (depending on total CM available and assigned) may be held in the CM queue. The Execute queue holds up to 7 queue jobs, competing in a round robin fashion, according to priority, for turns at control points.

CONTROL POINTS. Up to 15 available to user jobs, contain all jobs operating in a simultaneous, multiprogramming mode of execution (execution can only be accomplished from a control point). All central memory available for user jobs is controlled and assigned by the control points. A control point is not a physical entity but rather a concept used to facilitate bookkeeping. The control point number and control point area, however, are physical quantities that do appear in the system (Ref 17: Chapter 2-1). While only one job may be assigned to a control point, system resources such as central memory pointers, channels, equipment and processors required for the job may additionally be assigned to the control point Ref 17: Chapter 2-2).

Storage assigned to a single control point is contiguous although storage for all control points is not necessarily contiguous. Figure 3 is an example of how the control points and associated memory may appear. In addition to the 15 control points used for running jobs, two pseudo control points, numbered zero and 16, are used by the system.

Control point zero is used to identify system resources such as system libraries, pointers, and queue tables, not allocated to a job at a control point (Ref 17: Chapter 2-8). It provides a centralized location for multiple access modules, allowing them to perform functions for one or more jobs at their control points. This permits overall reduction of central memory usage - instead of several jobs having duplicate copies of these modules in their field lengths, only one set of these modules needs to occupy central memory (Ref 17: Chapter 2-18).

Control point 16 contains the Record Block Table RBT) containing information on each record block in a mass storage device. The RBT is file oriented consisting of work pairs which are linked to form an RBT chain for each file that exists on an allocatable device currently recognized by the system. A maximum of 8192 CM words are assigned

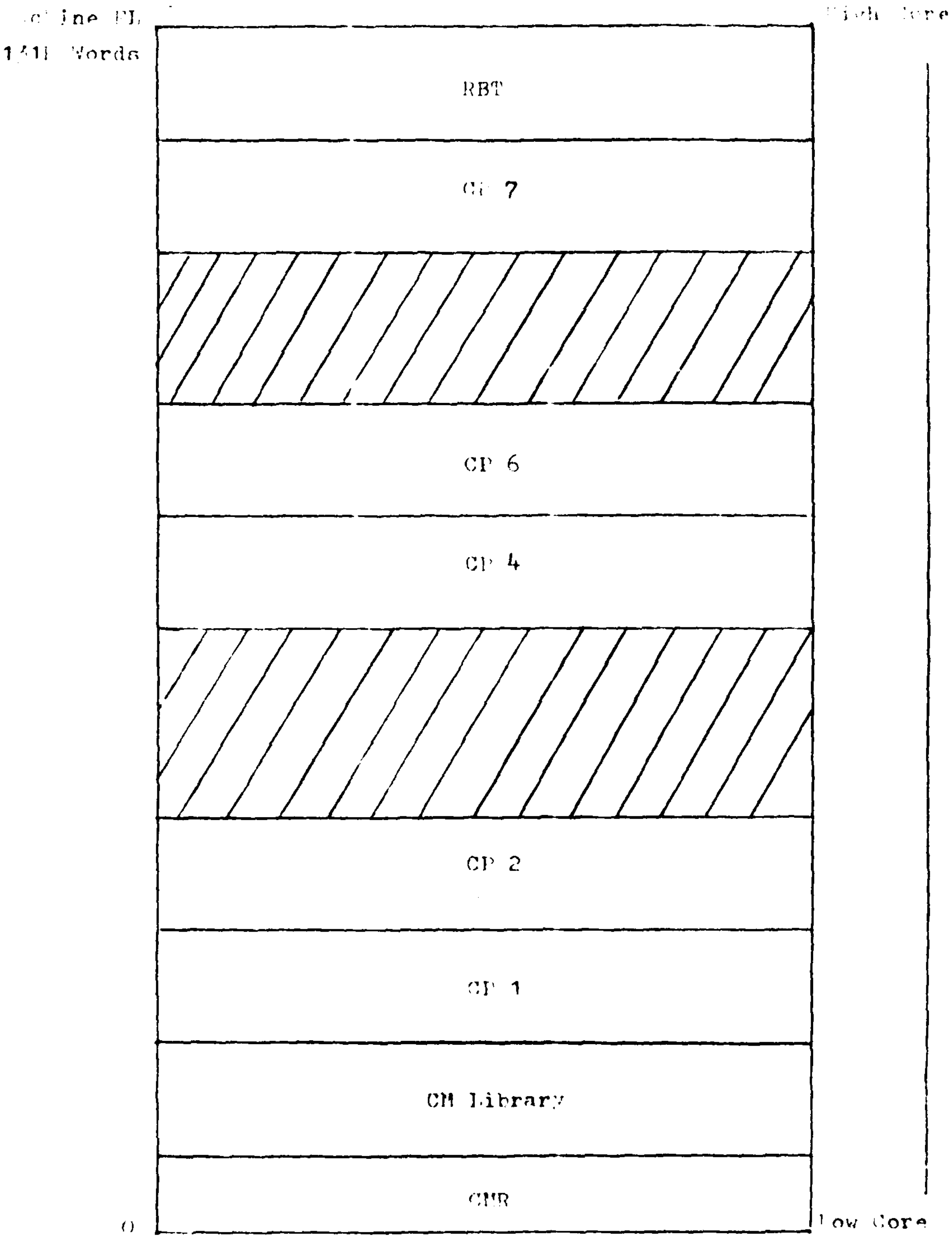


Fig. 5. Control Point/Memory Association

to contain all the RBT entries active at one time (Ref 17: Chapter 4-21).

All other control points are used for active user jobs. During the course of execution a job might remain continuously at the same control point (when few jobs are in the system or when locked into the control point by operator intervention). But it is more likely that a job will be swapped out while it is only partially executed. When the job is swapped back in it is probably associated with a control point other than the control point it previously vacated. Additionally, if, in the diagram of Figure 3, control 1 needs more storage, it will be necessary for the monitor (JANUS) to invoke a control point monitor (CPMTR) to obtain the required storage. CPMTR must then move control point 2 upwards to free the space contiguous to control point 1. Added storage always extends the field length upward (Ref 17: Chapter 2-3).

SWAPOUT is the term used when a job is forced by CPMTR to vacate its control point when the job is: (1) waiting for CM, (2) waiting permanent file access or (3) waiting for a permanent pack (Ref 17: Chapter 7-8). Once swapped, both the control point and central memory are released and made available for reassignment. The job, when swapped in, may be assigned a different control point and a different area in central memory (since all status tables and memory requirements associated with the job are swapped along with the job and not assigned to the control point previously used).

ROLLOUT occurs when a job (1) has non-allocatable equipment assigned, (2) is in a device waiting queue or (3) is in the operator queue waiting a response to some request. When rolled out, the job's field length and all of its memory is released. Its control point, however, remains assigned to the job while it waits in the queue for releasing. When rolled back in, the job begins active use of its control point receiving priority over the other queue jobs or a job currently executing at that control point (Ref 17: Chapter 7-7).

SCHEDULING

OVERVIEW. The SCOPE operating system controls the selection of jobs for execution from the input queue and the priority of jobs being processed. The scheduling algorithm may swap jobs into or out of execution or initiate new jobs with higher priority from the input queue. Both the availability of resources and the priorities of jobs are considered in the scheduling process. The criteria for job priorities, are, in part, set by the priority parameter (P) on the job card and by scheduler parameter variables which are set by installation management through deadstart procedures (Ref 14: vii).

Optimum scheduling for the CYBER workload has been established according to a representative job mix of the normal workload over a period of years. The software control group at AFLC (LMT) has set the scheduler parameter variables to achieve a maximum balance between system throughput, resource utilization, and job turnaround, although most of the parameters can be changed by the operator at the console. However, improvements in one area of performance through operator changes at the console may degrade another area to a non-acceptable level. Hence, both job card priorities and parameter changes from the console are discouraged.*

PRIORITIES used in the scheduling process are associated with a job but do not exist for the job continuously. They are assigned or evaluated at different stages of a jobs life. Figure 4 illustrates the assignment of these priorities at various stages of job processing. Basically, CDC has defined seven priority classes, 0 through 6, in its standard system.

Class 0 (initiation) - applies to all initiated batch jobs, regardless of their non-allocatable device requirements. (Non-allocatable devices include tapes and direct access Extended Core Storage (ECS), but are limited to tapes at

*Interview with LMO and LMT management personnel, May 1981.

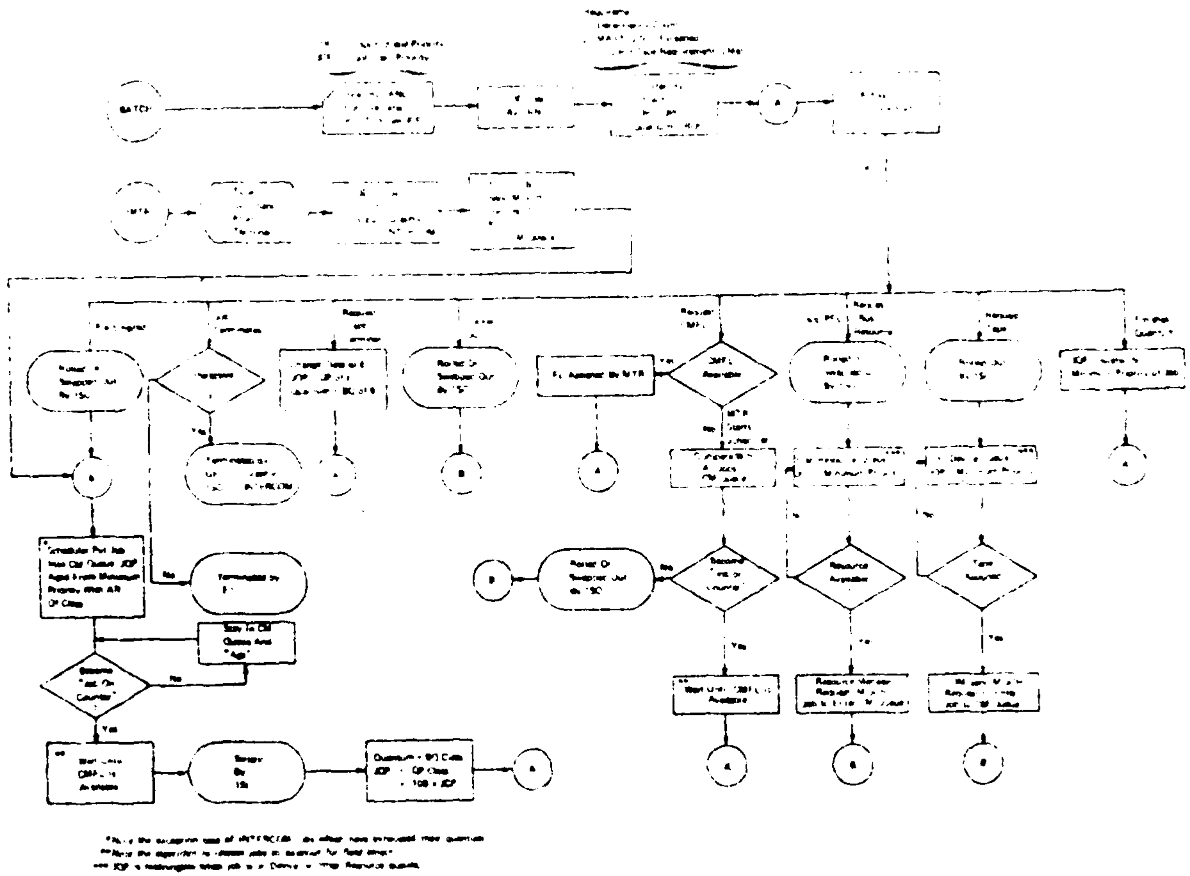


Fig 4. CYBER Scheduling Algorithm (Ref 14)

AFLC since ECS is not supported). This class controls the first time quantum used by each batch job. Once the job is swapped out, it will be placed in either class 1 or 2 depending on its resource requirements.

Class 1 (batch) - includes all batch jobs which do not require non-allocatable devices in order to execute.

Class 2 (device) - includes all batch jobs which require non-allocatable devices to satisfy their execution requirements.

Class 3 (intercom) - includes all interactive jobs or commands processed by INTERCOM.

Class 4 (multi-user) - jobs which are similar to intercom class jobs and servce several interactive users. Currently, EDITOR is the only standard multi-user job provided with the system.

Class 5 (express) - includes all jobs which require immediate processing to satisfy their requirements. These include:

(1) Jobs directed from the input queue by the operator.

(2) Jobs with fixed priorities (these are priorities directed by job card parameters).

(3) Jobs that are unlocked using the unlock EXP command from the console.

(4) Jobs that are dropped, killed or rerun (these are jobs requiring immediate routing from the system).

Class 6 (graphics) - this feature is not supported by AFLC.

Although installations may increase this number to 12, AFLC has found that four classes (1, 2, 3 and 5), along with 0, is best suited for their job flow purposes (Ref 14: Chapter 1,8).

In addition to initiation priorities, *scheduler priorities* required to accomplish optimal multiprogramming are defined below (Ref 14: Chapter 1,5-9).

Internal Job Card Priority (JCP) is a factor of the job class, its job card priority, and an aging factor. This factor is weighted and used as a factor in the computation of the job's job queue priority (JQP).

Job Queue Priority (JQP) is used when a job is competing with other jobs for central memory. This priority is a combination of factors, including class type, and is associated with the job throughout its execution.

Minimum Queue Priority (MINQP) is the level assigned to jobs first entering the central memory queue, or to a job which has used its allotted time slice (quantum). This factor is added to the weighted job class priority to obtain the JQP.

Maximum Queue Priority (MAXQP) is the maximum priority a job in the CM queue may achieve while waiting for scheduling. A job reaches this state by aging the MINQP.

Aging Rate (AR) is a factor used to weight the priority of a job according to the time it has spent in the CM queue.

Quantum Priority (QP) is the priority level assigned to a job once it has been swapped in. Here again, the JCP is weighted and added to the QP to obtain the job queue priority. MAXQP must be set at a lower value than QP to prevent jobs in the CM queue from pre-empting jobs before they have an opportunity to use their time slices (thrashing).

Base Quantum (BQ) is a measure of time that a job once at a control point, will maintain a priority equal to QP. That is, PP activity is weighted as one quarter of the CP time so that jobs performing considerable I/O will not have all of the slower PP activity charged against CP time.

Maximum Number of Jobs Initiated (MAXN) is a parameter used to determine the total number of jobs which can concurrently run at any given time. Thus, the total of all jobs, batch or device type, may not exceed the MAXN of that type except for express type jobs, or jobs which have been pre-aborted because the system could not satisfy all of the job's requirements (see tape scheduling).

JOB FLOW. (Ref Figure 4). When resources become available, the scheduler is initiated at one of two entry points. CP.SCH1 (entry point A in the diagram) is entered when a monitor cannot satisfy storage increase requirements for a job in central memory.

If, after finding a candidate job (see Best Job Selection) the requirements still cannot be met, the scheduler enters CP.SCH (entry point B in the diagram). The scheduler may also enter CP.SCH by initiation from one of the following:

When a PP routine requests scheduler

When the Field Length (FL) assigned to a job is reduced

When scheduler's periodic recall period has expired

When scheduler's request stack is full

Once entered, the scheduler will first try to initiate jobs from the input queue. If none are available or MAXN is reached, jobs in the CM queue which are candidates for swap-in are selected by priority.

Best Job Selection. The scheduler, once initiated, selects the best candidate meeting all of the following:

(1) The job's CM field length requirements do not exceed available FL + psuedo available FL (This is the total field length of all jobs swapped out or terminating) + the sum of the FL of the jobs in CM which have a lower job queue priority.

(2) The job must have the highest job queue priority among jobs competing for CM assignment.

(3) The job cannot be locked out.

If two jobs meet the requirements, the job requiring the larger field length will be selected first. If no job can be selected, the scheduler drops out and repeats the process on the next reentry.

Swap-Out Job. Once the scheduler finds a candidate job for execution it checks to see that there is enough available field length for the job. If there is enough available, the scheduler initiates a swap-in. If the field length is insufficient, the scheduler re-accomplishes its "best job" procedure to find another candidate. On the second try, if FL is still insufficient, the scheduler initiates a swap-out of the lowest priority job, regardless of whether that job's quantum has completed. Enough jobs are swapped/rolled out to permit the selected job to acquire its required field length and execute.

Swap-In Job. Jobs which are candidates for swap-in, as determined through the "best job" selection procedure, are either (1) assigned to the next available control point, if one exists, (2) assigned to the control point of the first lower priority executing job which can free up enough field length that, when coupled with available FL, can meet swap-in's requirements, or (3) is returned to the CM queue. If a swap-in cannot be accomplished at this point, the scheduler returns to the Best Job Procedure but this time to consider only rolled out jobs for selection as candidates.

Termination. Once a job has completed all of its execution, its resources are returned and an output priority is assigned. The job waits in the output queue until the output resources required are satisfied. Again, as with input queue priorities, an Aging Rate factor is applied to jobs waiting for output.

TAPE SCHEDULING is designed to improve overall system throughput, particularly as it relates to tape job setup and execution (Ref 14: Chapter 4,25-26). All incoming tape jobs are entered in a pre-scheduling queue, a subset of the input queue, to allow the operator some control over the selection of tape jobs for execution. The drive assignment itself is based on an overcommitment algorithm which assumes that a job does not always need its maximum tape requirements for the full duration of the job and

that most processing activity uses less than the maximum number of drives necessary for job execution.

(Job execution in this sense means a particular job step execution. That is, a job may require, for example, four tapes for the entire process but only one or two for any single job step. (e.g., step 1 - compile using tape 1, step 2 - execute using tapes 2 and 3, step 3 - sort using tape 4). Therefore, if the job requests all four tapes for step 2 instead of just prior to the step requiring the tape, serious degradation of efficiency and multiprogramming may occur, especially if step three occurs 10, 20 minutes or longer after step one.)

Therefore, a job is assigned only those drives it needs to continue execution at any instant in time. Excess drives, at that instant, are made available to run other jobs. Such a job scheduling scheme permits the total tape requirements of all active jobs to exceed the total number of drives in the installation and may cause a potential deadlock by allowing two or more jobs to have unfilled tape demands such that every available tape drive is assigned to a job but no job has enough tapes to run to completion.

SCOPE includes some deadlock prevention features by not acknowledging a REQUEST when only one drive is remaining and several tape jobs are in various states of execution. However, should another REQUEST be issued from a different job, the deadlock may occur and the operator must take action by causing a rerun of one of the tape jobs, or by killing a job which is using a requesting tape, or, if possible, wait for still another tape job to complete.

MANAGEMENT ORGANIZATION

Organizationally, five directorates share responsibility for the management of the system. Two, the Directorate of Technical Support (LMT), and the Directorate of Operations (LMO), both have primary responsibility with LMT performing the functions of determining what software is to be acquired, and developing, maintaining, and assuring software support for the system while LMO has physical responsibility for operations, scheduling, and maintenance of the computer. Two user directorates, LMV and LMZ, supply the basic workloads for the system while the fifth, LMD, does all outside contracting for hardware, software, and support equipment acquisitions.

Basically, both LMO and LMT attempt to respond to the users needs from their areas within the constraints of the current machine configuration and software support available. The computer is run under a closed shop environment and scheduled with an 18 hours workload six days per week. The system itself is operational 24 hours per day seven days per week which includes maintenance scheduled for 3 four-hour periods per week. Figure 5 is a typical weekly schedule showing times reserved for software testing, maintenance, and planned-idle time.

In the technical support area, LMT personnel have been aiding the users by providing technical expertise in recommending programming methods, where possible, which take advantage of the increased capabilities of the CYBER over the older 3080 computer. Additionally, a recent study made by LMT supporting a Data Automation Requirement for newer and faster peripheral equipment, indicates that a different hardware configuration may provide some relief for the current throughput problem (Ref 1:1).

Figure 6 graphically portrays the potential monthly workload for the computer after the remaining conversion systems are added to the job mix with no planned change in the system's configuration.

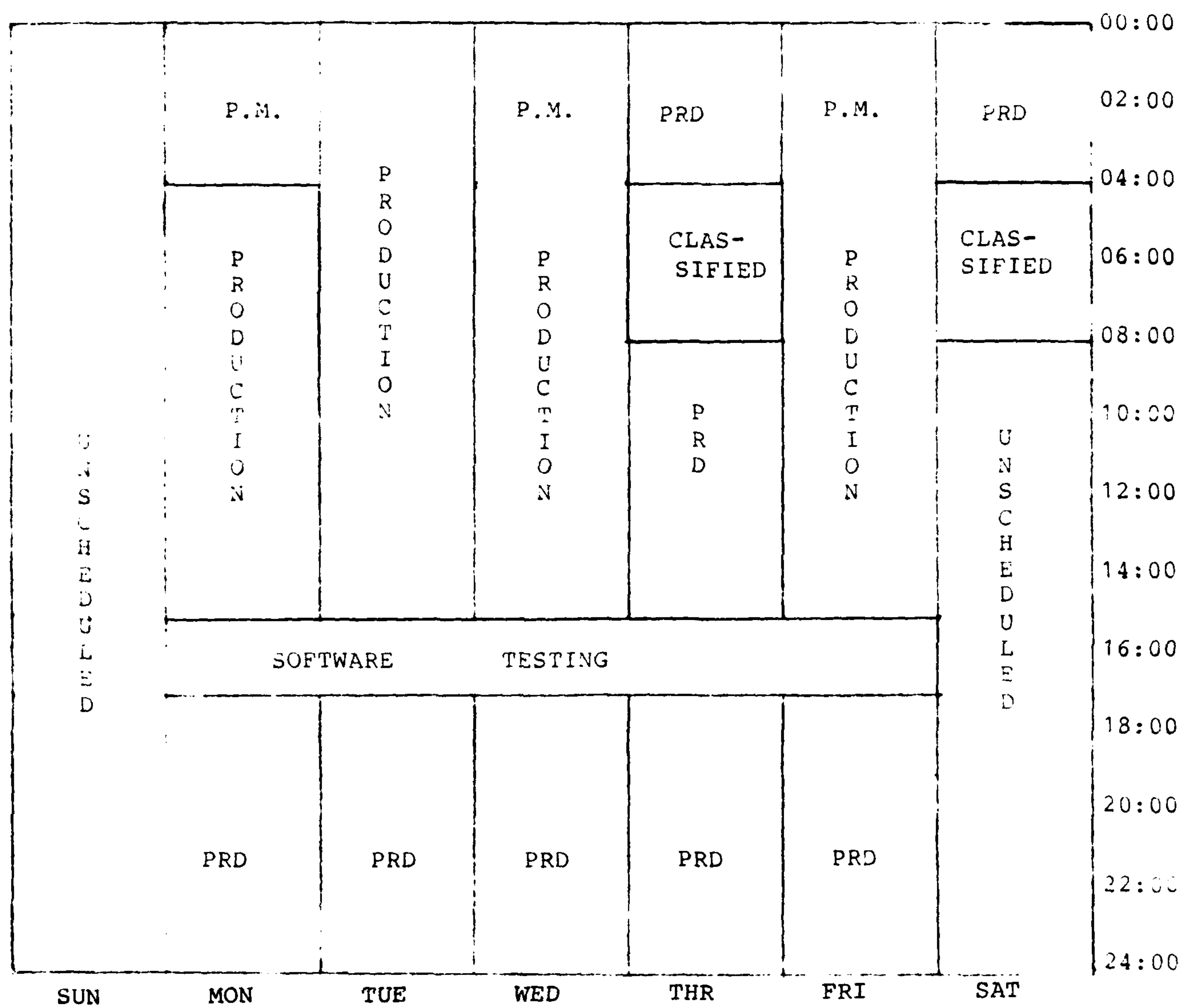


Fig 5. Typical CYBER Weekly Schedule

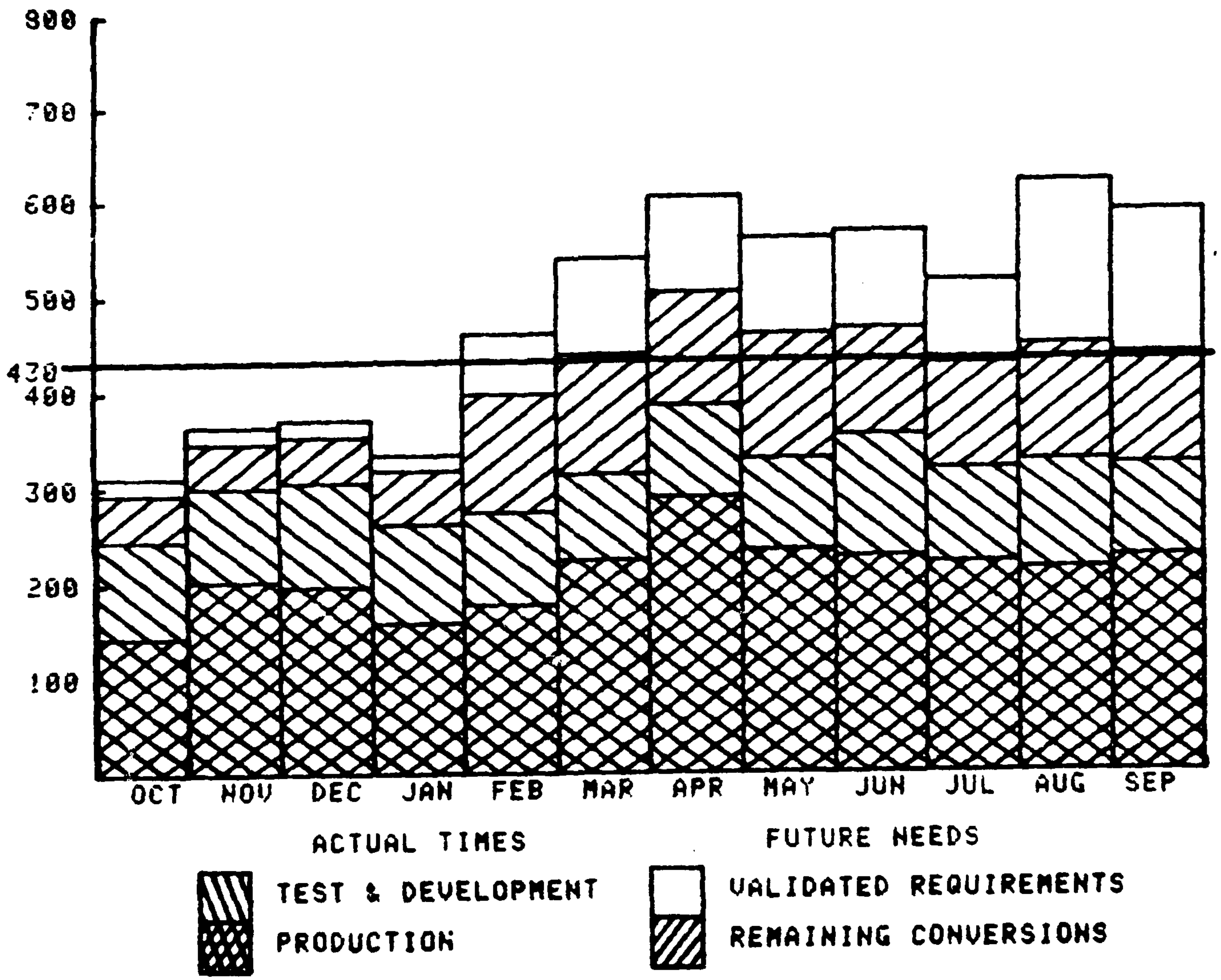


Fig 6. HQ AFLC CYBER 73
CYBER PROCESSING HOURS
1980-1981

IV WORKLOAD CHARACTERISTICS AND BENCHMARK SELECTION

WORKLOAD CHARACTERISTICS

The philosophy discussed in Chapter II for characterizing the CYBER operations was used to collect and parameterize 23 characteristics of a job, mapping an audit trail of its flow through the system. The production master schedule (Ref: 30) was analyzed to determine if stability existed in the normal day-to-day operations. Additionally, in-depth discussions were held with the machine schedulers and operators to verify the consistency of the actual job entry into the computer. A determination was made that during actual production some jobs were given priority over others through a method of "locking in" jobs from the console to a control point until completion. This tends to lessen the effect of the system scheduling algorithm when a full mix of jobs is waiting in the input queue (Ref 17:Chapter 7,7). The practice was done primarily on sort jobs where a high correlation was perceived to exist between CPU and I/O activity, and during lightly scheduled periods between 1800-0600 weekdays and on weekends. The effect degrades multiprogramming but does not tend to hinder the installation's final output requirement since these periods are lightly scheduled as a practice.

Because of the heavy demand for program/system testing and overall production output requirements during prime hours (0800-1600), locking activities during this interval were, for the most part, non-existent. Furthermore, specific periods were set aside for in-house operations (e.g., maintenance, software development and testing, and unscheduled time) (Ref: 30). Therefore, based on the information above and the results of a T-TEST on a sampling of data from the weeks of May 4-10 vs July 6-12, which

indicated, at the .10 level, that no statistical difference existed in the Jobs processed, a cycle was determined to exist based on daily, weekly and monthly interations of user jobs. Thus, aside from testing, which is tied to future workloads which vary by month, any month in which the full schedule of jobs were processed could serve as a baseline population for the benchmark selection.

To this end, accounting data consisting of DAYFILE data from four weeks of July. 1981, (excluding the week of July 13-19 which was not available) was provided by the AFLC Data Operations Directorate (LMO) for use in the analysis. Over 468,000 transactions were run through the series of reduction programs (see Chapter II) providing 5,064 transactions representing statistically the workload of user jobs for the month. The reduction series was then followed by a sequence of eight SPSS (Ref:34) data manipulation runs in order to stratify the workload variables for the SPSS data analysis procedures referred to in Chapter II. Appendix B contains the procedures and methods used along with the complete Job Control Language (JCL) necessary to re-accomplish the task.

SYSTEM WORKLOAD

Table 6 shows SPSS Continuous Data Descriptions (CONDESCRIPTIVE) of the July workload. These data exclude in-house operations such as deadstarts, reruns and restarts since these are considered as overhead of operations and not controllable (Ref 20:228).

Table 7, summarizing the total machine activity, including in-house operations, is an extract from the CLARA (Ref:11) outputs of resource usage, and is further portrayed in Figure 7.

TABLE 6

MEAN VALUES OF WORKLOAD VARIABLES

VARIABLE	DESCRIPTION	MEAN MINUTES/WORDS
ARRIVAL	TIME OF JOB ARRIVAL	11.57
INQUEUE	TIME IN INPUT QUEUE	6.25
IRRIVAL	INTER-ARRIVAL RATE	6.58
REQ7	7 TRACK TAPES REQUESTED	[illegible]
REQ9	9 TRACK TAPES REQUESTED	1.20
TREQ	TIME REQUESTED	3[illegible].99
CMREQ	CM WORDS REQUESTED	62K
DISKWAIT	TIME WAITING FOR DISK	1.31
TAPEWAIT	TIME WAITING FOR TAPES	1.24
DISKIO	NUMBER OF DISK ACCESSES	7K
TAPEIO	NUMBER OF TAPE BLOCKS I/O	153K
SORT	TIME SPENT SORTING	.96
SORTED	NUMBER OF RECORDS SORTED	64K
SEVENTRK	NUMBER OF 7 TRACK TAPES USED	1.98
NINETRK	NUMBER OF 9 TRACK TAPES USED	1.57
CORE	AMOUNT OF CORE USED	56K
CPU	CENTRAL PROCESSOR TIME	2.45
CM	CENTRAL MEMORY TIME	5.08
IO	I/O TIME USED	2.09
PPUSAGE	PERIPHERAL PROCESSOR TIME	4.66
LINES	LINES PRINTED	1.66
POSTPROC	POST PROCESSING TIME	6.53
THRUPUT	SYSTEM THRUPUT TIME	11.97
TURNAROUND	TURNAROUND TIME	24.74

TABLE 7

RESOURCE UTILIZATION FOR JULY

WEEK	TOTAL JOBS	PERCENT JOBS IN ABNORMAL TERM	JOBS PRO-CESSED PER HOUR	MULTI PROGRAM-MING	PERCENT CONTROL POINT USAGE	PERCENT CENTRAL MEMORY USAGE	PERCENT DISK USAGE	PERCENT CPU USAGE
1	1218	6.00	11.80	4.95	38.43	48.84	46.58	53.35
2	2023	8.97	13.21	5.63	32.89	73.70	64.69	50.67
3	1292	10.83	9.85	3.73	36.62	43.02	45.40	40.05
4	1646	9.51	10.31	4.47	32.70	45.47	47.04	41.31
5	909*	14.30	10.93	2.64	39.84	46.98	54.70	45.70
ALL	7088	9.92	11.22	4.28	36.10	51.60	51.68	46.22

Source: CLARA (Ref:11)

*Does not include last day of month

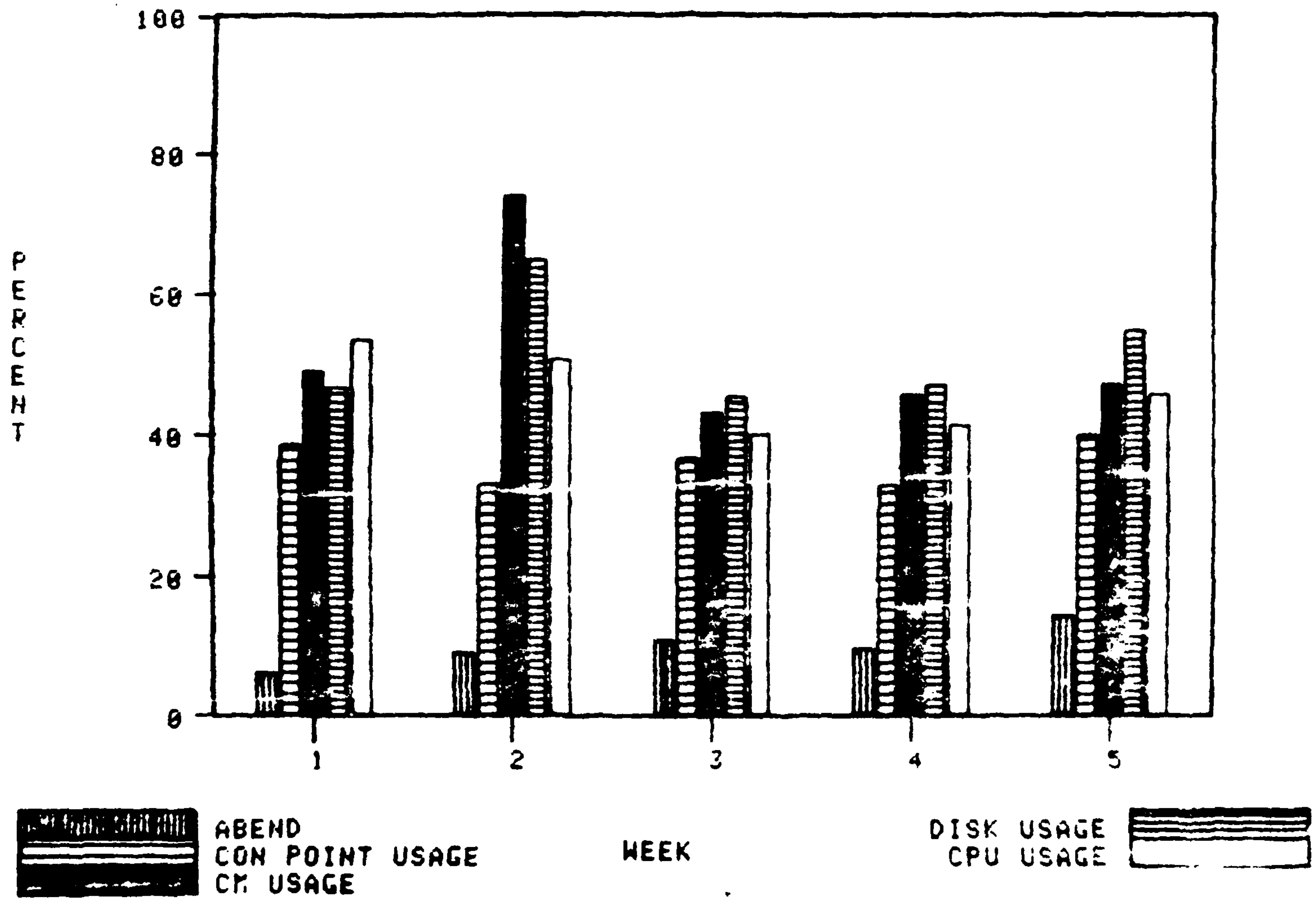


FIG 7. MONTHLY RESOURCE UTILIZATION
*SOURCE CLARA Ref 11

BENCHMARK

In attempting to determine which set of samples to use for a benchmark, a T-TEST was run against weekly groups to test whether a difference existed from week to week in the MEAN values of ARRIVAL time, CPU usage, I/O usage, SWAP time and TURNAROUND.

1055, 1723, 1343 and 943 samples were used representing respectively, four weeks of July data. The week of July 13-19 was unavailable and not used in the test. A null hpyothesis (Ho) was established that no difference existed between the weekly processing. If not rejected, then any randomly selected week could serve as a benchmark. The alternative (Ha) would establish that a significant difference existed statistically and another approach to selecting the benchmark was necessary. The basis for decision was: If the T-Value of the test exceeds to T-Value at the .05 Level of Significance (34:267-275), reject the null and accept the alternative.

As seen from the results (Table 8), the hypothesis that any one week provides a good representation of the monthly workload must be rejected. For this reason, the full monthly sample database is used as the Benchmark Workload (Table 6).

TABLE 8

TEST OF DIFFERENCES IN

WEEKLY PROCESSING

WEEKS COMPARED	VARIABLE	T-VALUE	.05 LEVEL	RESULT
1 vs 2	ARRIVAL	-0.62	-1.65	ACCEPT Ho
	CPU time	1.77	1.65	REJECT Ho
	I/O time	-1.37	-1.65	ACCEPT Ho
	SORTING	0.13	1.65	ACCEPT Ho
	TAPES used	1.17	1.65	ACCEPT Ho
	TURNAROUND	-0.51	-1.65	ACCEPT Ho
1 vs 5	ARRIVAL	2.91	1.65	REJECT Ho
	CPU time	1.14	1.65	ACCEPT Ho
	I/O time	-2.64	-1.65	REJECT Ho
	SORTING	-0.69	-1.65	ACCPET Ho
	TAPES used	0.34	1.65	ACCEPT Ho
	TURNAROUND	-3.81	-1.65	REJECT Ho
2 vs 4	ARRIVAL	2.16	1.65	REJECT Ho
	CPU time	-0.55	-1.65	ACCEPT Ho
	I/O time	-0.37	-1.65	ACCEPT Ho
	SORTING	-0.39	-1.65	ACCEPT Ho
	TAPES used	-1.66	-1.65	ACCEPT Ho
	TURNAROUND	-0.91	-1.65	ACCEPT Ho
4 vs 5	ARRIVAL	-1.90	-1.65	REJECT Ho
	CPU time	-0.01	-1.65	ACCEPT Ho
	I/O time	-0.97	-1.65	ACCEPT Ho
	SORTING	0.96	1.65	ACCEPT Ho
	TAPES used	0.62	1.65	ACCEPT Ho
	TURNAROUND	-3.02	-1.65	REJECT Ho

*SOURCE: SPSS T-TEST (Ref 34:267-275)

WORKLOAD ANALYSIS

Tables 9 thru 17 are SPSS FREQUENCIES output tables of the parameters used for the analysis. The following are initial observations about the data which provides the basis for the development of hypotheses about possible thruput bottlenecks.

TIME OF ARRIVAL. This variable, based on two hour increments presents a relatively uniform distribution with peak activity occurring around the 10:00 AM period. Of the 5,064 samples taken, 20.5 percent are arrivals prior to the prime shift (0800-1600), 26.1 percent occur after the prime shift and the remaining 53.4 percent occur during the prime period (Figure 8A).

INTER-ARRIVAL RATE. The time between job arrivals is one of the more interesting statistics generated. Stratified in two-hour increments and protrayed on a monthly and weekly basis (Figures 8B-C), the mean *inter-arrival rate* can be compared with the mean turnaround time for the month (Table 6). The monthly mean values show an average turnaround of 24.74 minutes per job while the arrival time follows an exponential distribtuion with 6.58 minutes between jobs, or 1/6.58 = .15 jobs per minute. At that rate, it appears that the machine cannot keep up with the workload. The comparison is misleading, however, since the inter-arrival rate does not account for the multi-programming capability of the machine. To get a feel of the effective inter-arrival rate under multi-programming, the mean multi-programming rate given in Table 7 was used to calculate a weighted effective inter-arrival rate for the periods measured.

EFFECTIVE RATE = IARRIVAL * (MULTI-PROGRAMMING/
SAMPLE JOBS/TOTAL JOBS) (eqn 1)

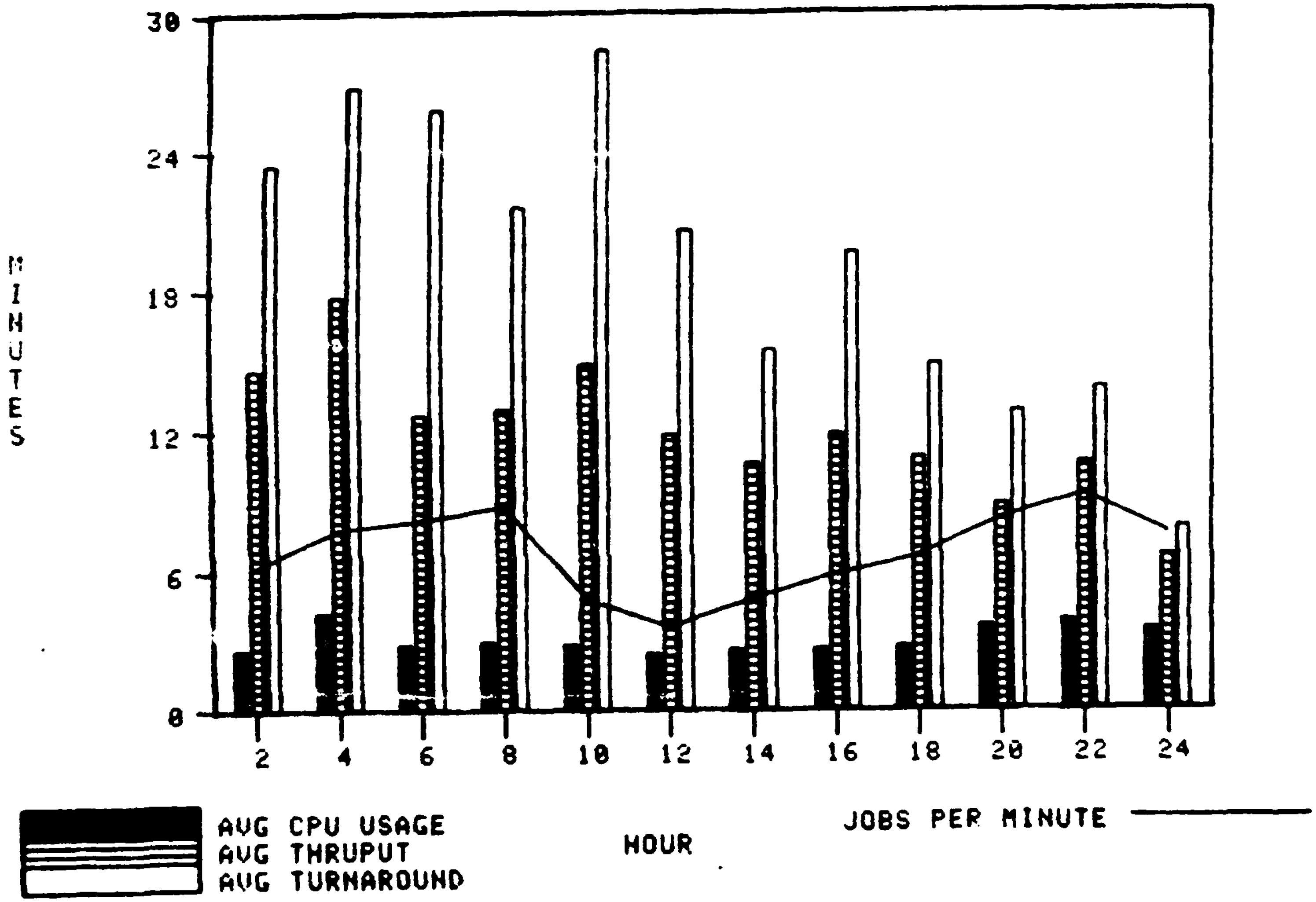


FIG 8a. HOURLY TURNAROUND
in relation to Inter-arrival time
(MONTHLY)

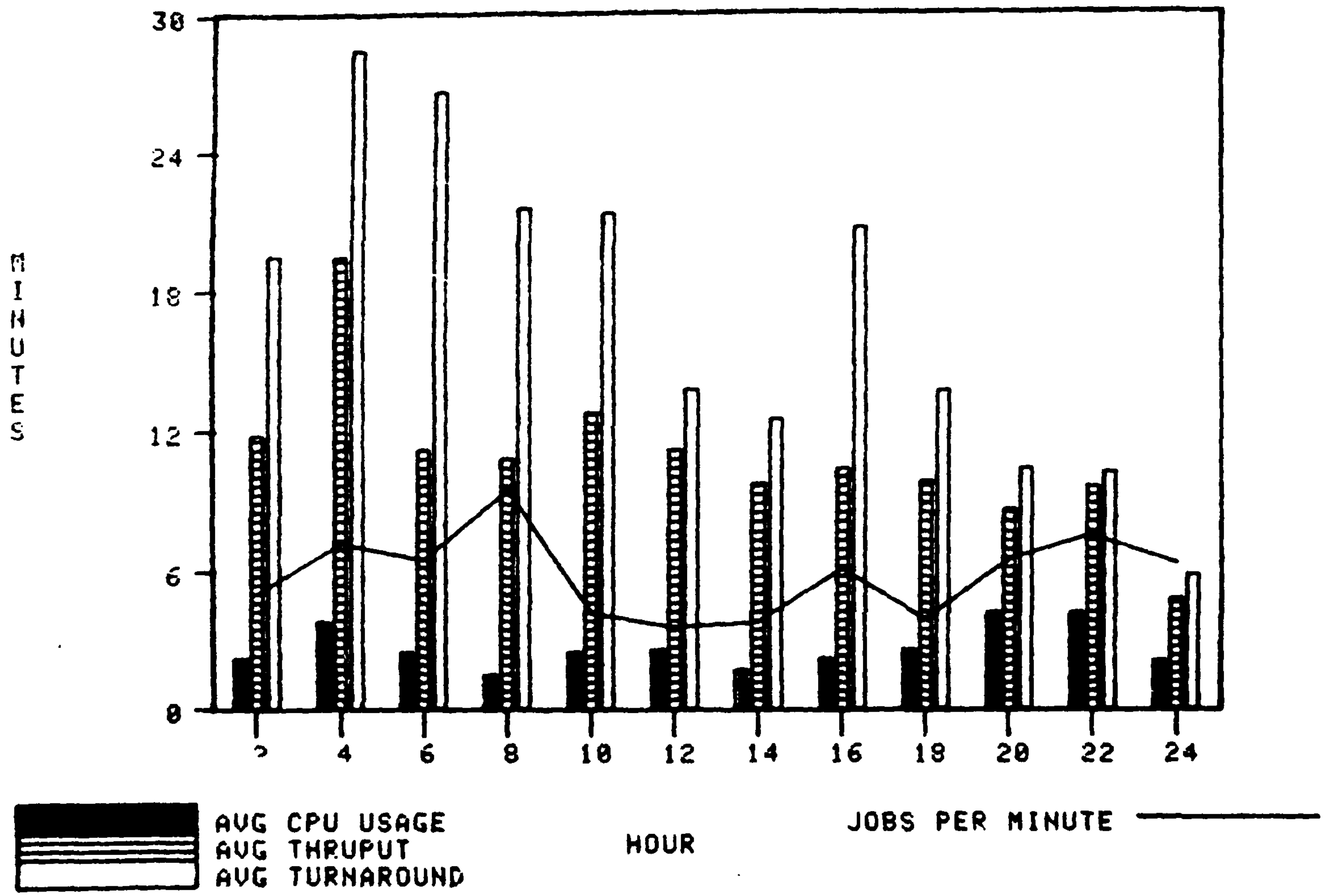


FIG 8b. HOURLY TURNAROUND
in relation to Inter-arrival time
(WEEKLY)

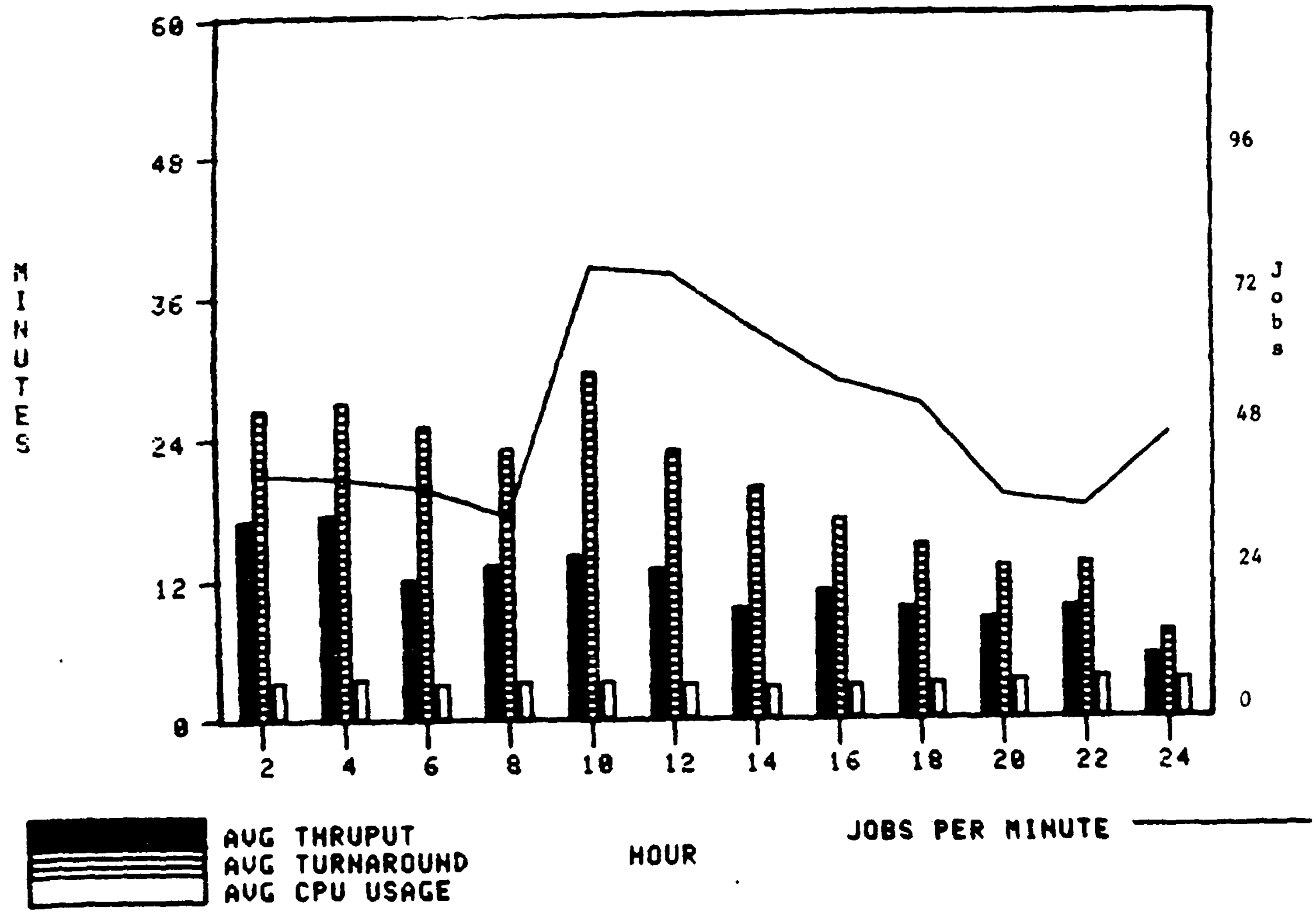


FIG 8c. HOURLY TURNAROUND
in relation to Job Arrival
(July 6-12)

TABLE 9A
TIME OF JOB ARRIVAL

CATEGORY	FREQUENCY	PERCENT	CUM (PCT)
0000-0200	356	7.0	7.0
0200-0400	354	7.0	14.0
0400-0600	325	6.4	20.5
0600-0800	275	5.4	25.9
0800-1000	680	13.5	39.4
1000-1200	668	13.2	52.6
1200-1400	567	11.2	63.8
1400-1600	462	9.1	72.9
1600-1800	488	9.6	82.5
1800-2000	288	5.7	88.2
2000-2200	287	5.7	93.9
2200-2400	314	6.1	100.0
TOTAL	5064	100.0	

TABLE 9B
INTER-ARRIVAL RATE

CATEGORY	FREQUENCY	PERCENT	CUM (PCT)
LESS THAN 2 MINUTES	2543	50.2	50.2
2 TO 4 MINUTES	811	16.1	66.3
4 TO 6 MINUTES	496	9.8	76.1
6 TO 8 MINUTES	259	5.1	81.2
8 TO 10 MINUTES	356	7.0	88.2
10 TO 12 MINUTES	35	.7	88.9
12 TO 14 MINUTES	17	.3	89.2
14 TO 16 MINUTES	18	.4	89.6
16 TO 20 MINUTES	16	.3	89.9
MORE THAN 20 MINUTES	511	10.1	100.0
TOTAL	5064	100.0	

TABLE 9C

INTER-ARRIVAL/TURNAROUND TIMES FOR JULY

DAY	NUMBER OF ARRIVALS	INTERARRIVAL RATE (in minutes)	TURNAROUND TIME (in minutes)
1	214	6.56	13.21
2	309	4.55	36.64
3	166	8.52	8.85
4	147	9.54	6.81
5	219	6.41	18.44
6	221	6.37	28.71
7	318	4.44	15.25
8	267	5.29	20.68
9	305	4.62	18.61
10	297	4.71	15.09
11	195	7.19	51.12
12	120	11.62	8.85
20	219	6.37	19.12
21	289	4.89	18.00
22	270	5.19	11.75
23	284	4.89	14.38
24	150	9.30	38.58
25	67	20.14	31.49
26	64	22.02	22.41
27	137	10.26	35.72
28	194	7.12	26.93
29	137	8.43	36.90
30	248	5.63	38.82
31	227	6.19	36.13
MONTH	5064	6.43	24.74

The resulting value gives an effective inter-arrival rate of 27.76 minutes compared to the 24.74 minutes to start and complete a job.

TIME WAITING FOR DISK MOUNTING. This statistic, which will be used to test a hypothesis on insufficient devices and inadaquate channel distribution, is applicable for just 15 percent (747 jobs) of the samples taken. The majority of jobs (85.2 percent) had no wait time for disk mounting. This could be the result of either no disk requirement or the reading/writing of additonal files once the disk was mounted, or of jobs that used public devices for data manipulation. Of the 747 jobs, however, 6.6 percent waited less than a minute while 2.5 percent (131 jobs) waited nine or more minutes for a disk pack.

TIME WAITING FOR TAPE MOUNTING. Statistics for this category show a small average wait time. The statistic, however, is probably misleading as to the actual time a job waits for a tape drive. According to the tape scheduling algorithm (Chapter II), jobs requiring tape are first placed in a special wait queue until released by the operator. In this instance, an operator may hold the requesting job in the queue until a drive is available, thus preventing the full wait period from being known by the system. Hence, the statistics shown for tape waiting are not necessarily representative of the actual occurrence of waiting. The data, however, does indicate that on the average 1.71 minutes were used waiting for tapes once the job entered the input queue.

TABLE 10A

TIME WAITING FOR DISK MOUNTING

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
NO WAITING	4317	85.2	85.2
LESS THAN 1 MINUTE	341	6.6	91.3
1 TO 2 MINUTES	93	1.8	93.6
2 TO 3 MINUTES	50	1.0	94.6
3 TO 4 MINUTES	34	.7	95.3
4 TO 5 MINUTES	33	.7	96.0
5 TO 6 MINUTES	19	.4	96.4
6 TO 7 MINUTES	21	.4	96.8
7 TO 8 MINUTES	18	.4	97.2
8 TO 9 MINUTES	13	.3	97.5
MORE THAN 9 MINUTES	131	2.5	100.0
TOTAL	5064	100.0	

TABLE 10B

TIME WAITING FOR TAPE MOUNTING

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
LESS THAN .01 MINUTE	3689	72.8	72.8
.01 TO .1 MINUTES	80	1.6	74.4
.1 TO .5 MINUTES	324	6.4	80.8
.5 TO 1 MINUTE	235	4.6	85.4
1 TO 1.5 MINUTES	141	2.9	88.3
MORE THAN 1.5 MINUTES	595	11.7	100.0
TOTAL	5064	100.0	

The true wait time for tape jobs is possibly hidden in the overall turnaround time. This theory will be investigated when the hypotheses are evaluated.

SEVEN TRACK TAPES USED. Approximately 80 percent of the sample cases used no tape facilities according to this category and 13.4 percent used just one. Judging from the SEVENTRK Utilization Table (Table 11B), the daily utilization rate for July was 1.36:

$$\text{TOTAL JOBS} / \left(\sum_{i=1}^{9} \text{TAPES} * \text{FREQUENCY} \right) = \text{RATE} \qquad \text{(eqn 2)}$$

7 TRACK TAPES REQUESTED. Here again, a high percentage of jobs, 76.5 percent, requested no tape facilities while 16 percent requested one drive. Jobs requesting 3 or less tapes account for all but 1.8 percent of tape requests for the period. This, however, is 1.7 percent more requests than the same statistical number of jobs using 3 or less tapes. Moreover, according to Table 11A, on 7 TRACK REQUEST, the 1.7 percent amounts to 88 jobs requesting facilities which were not used.

9 TRACK TAPES USED. This statistic, much the same as with seven track tapes used, indicates that 95 percent of the samples used one tape or less. Similarly, the utilization rate turns out to be 2.94:

$$\text{TOTAL JOBS} / \left(\sum_{i=1}^{7} \text{TAPES} * \text{FREQUENCY} \right) = \text{RATE} \qquad \text{(eqn 3)}$$

9 TRACK TAPES REQUESTED. This category, closely aligned to similar 7 TRACK tape activity, shows 99 percent of all samples requested 3 or less 9 TRACK facilities. 26.8 percent requested some tape facilities, however, and this contrasts with the 18.1 percent of jobs actually using the resource. Again, according to the frequency tables (Tables 12 A & B), 447 jobs appear to have requested facilities which are not used.

TABLE 11A

7 TRACK TAPES REQUESTED

NUMBER OF TAPES	FREQUENCY	PERCENT	CUM (PCT)
0	3874	76.5	76.5
1	812	16.0	92.5
2	190	3.8	96.3
3	100	2.0	98.3
4	33	.7	99.0
5	36	.7	99.7
6	11	.2	99.9
7	8	.1	100.0
TOTAL	5064	100.0	

TABLE 11B

7 TRACK TAPES USED

NUMBER OF TAPES	FREQUENCY	PERCENT	CUM (PCT)
0	4054	80.0	80.0
1	680	13.4	93.4
2	104	2.1	95.5
3	70	1.4	96.9
4	37	.7	97.6
5	44	.9	98.5
6	18	.4	98.9
7	57	1.2	100.0
TOTAL	5064	100.0	

TABLE 12A

9 TRACK TAPES REQUESTED

NUMBER OF TAPES	FREQUENCY	PERCENT	CUM (PCT)
0	3705	73.2	73.2
1	1197	23.6	96.8
2	90	1.8	98.6
3	46	.9	99.5
4	21	.4	99.9
5	5	.1	100.0
TOTAL	5064	100.0	

TABLE 12B

9 TRACK TAPES USED

NUMBER OF TAPES	FREQUENCY	PERCENT	CUM (PCT)
0	4150	82.0	82.0
1	634	12.5	94.5
2	151	3.0	97.5
3	66	1.3	98.8
4	37	.7	99.5
5	26	.5	100.0
TOTAL 5064	100.0		

TURNAROUND. As shown in Table 13A, 41.6 percent of the samples completed processing in less than one minute while nearly two-thirds were finished in five or less minutes. In the high use category, 8 percent used an hour or more with 40 percent of those (4.3 percent of all jobs) processing more than two hours. The 74 jobs which processed more than 6 hours amount to an average 3.08 jobs per day processing for more than 25 percent of the available time.

TIME FOR ALL I/O FUNCTIONS. I/O time used per job averaged 2.4 minutes with more than half of all jobs using less than 1 minute. 6.2 percent of the samples required more than 10 minutes including .8 percent (30 jobs) which required an hour or more of I/O processing.

DISKIO. In this sub category of I/O processing, 65.5 percent made less than 1,000 accesses while another 31 percent made from 1,000 to 40,000 accesses. 3.5 percent, or 179 jobs, accessed the disk 40,000 times or more.

TAPEIO. This subcategory was less pronounced than DISKIO. 77.1 percent of all jobs had no tape requirements while 12.5 percent used uniform increments of 5,000 blocks in the 1,000 to 200,000 range. More conspicuous are the 548 jobs (10.5 percent) requiring more than 200,000 tape writes.

TABLE 13A

TURNAROUND TIME

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
LESS THAN 1 MINUTE	2106	41.6	41.6
1 TO 5 MINUTES	1012	20.0	61.6
5 TO 10 MINUTES	413	8.2	69.8
10 TO 20 MINUTES	319	6.3	76.1
20 TO 30 MINUTES	169	3.3	79.4
30 TO 60 MINUTES	590	11.7	91.1
60 TO 90 MINUTES	153	3.0	94.1
90 TO 120 MINUTES	85	1.7	95.8
2 TO 3 HOURS	86	1.7	97.5
3 TO 6 HOURS	57	1.1	98.6
MORE THAN 6 HOURS	74	1.4	100.0
TOTAL	5064	100.0	

TABLE 13B

THRUPUT TIME

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
LESS THAN 1 MINUTE	2621	51.8	51.8
1 TO 5 MINUTES	1052	20.8	72.6
5 TO 10 MINUTES	370	7.3	79.9
10 TO 20 MINUTES	231	4.5	84.4
20 TO 30 MINUTES	150	3.0	87.4
30 TO 60 MINUTES	418	8.3	95.7
60 TO 90 MINUTES	90	1.7	97.4
90 TO 120 MINUTES	49	1.0	98.4
2 TO 3 HOURS	44	.9	99.3
3 TO 6 HOURS	25	.5	99.8
MORE THAN 6 HOURS	14	.2	100.0
TOTAL	5064	100.0	

TABLE 14A

TIME FOR ALL I/O FUNCTIONS

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
NONE	1076	20.6	20.6
LESS THAN 1 MINUTE	2986	57.2	77.8
1 TO 5 MINUTES	615	11.8	89.6
5 TO 10 MINUTES	249	4.8	94.4
10 TO 30 MINUTES	212	4.1	98.5
30 TO 60 MINUTES	51	1.0	99.5
MORE THAN 1 HOUR	30	.5	100.0
TOTAL	5064	100.0	

TABLE 14B

NUMBER OF DISK ACCESSES

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
Less Than 1,000	3318	65.5	65.5
1,000 TO 5,000	798	15.8	81.3
5,000 TO 10,000	283	5.6	86.9
10,000 TO 20,000	291	5.7	92.6
20,000 TO 40,000	195	3.9	96.5
40,000 TO 60,000	62	1.2	97.7
MORE THAN 60,000	117	2.3	100.0
TOTAL	5064	100.0	

TABLE 14C

TAPE BLOCKS READ/WRITTEN

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
NO TAPE I/O	4022	77.1	77.1
LESS THAN 1,000	196	3.8	80.9
1,000 TO 5,000	89	1.7	82.6
5,000 TO 10,000	75	1.4	84.0
10,000 TO 20,000	83	1.6	85.6
20,000 TO 50,000	77	1.5	87.1
50,000 TO 100,000	66	1.3	88.4
100,000 TO 150,000	31	.6	89.0
150,000 TO 200,000	31	.6	89.6
MORE THAN 200,000	548	10.4	100.0
TOTAL	5064	100.0	

SORTING. Although 88.9 percent of the jobs processed accomplished no sorting, nearly 5.2 percent of those that did, sorted from 10,000 to 300,000 records, and 35 jobs sorted in excess of 300,000. These 35 jobs amount to slightly more than one job per day. Furthermore, the statistics indicate that 3 jobs used over an hour for sorting data (see Table 15B).

CORE USAGE. Of the CORE blocks used, more than two-thirds (68.7 percent) used less than 10,000 CORE positions with 27.1 percent using between 10,000 and 200,000 positions. More worth noting are the 361 (6.9 percent) jobs which used more than 250,000 positions of core. These jobs average 11.6 per day, and, if found uniform throughout the period, could have grave effects on multiprogramming.

TABLE 15A

RECORDS SORTED

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
NONE	4641	88.9	88.9
LESS THAN 1,000	137	2.6	91.5
1,000 TO 5,000	94	1.8	93.4
5,000 TO 10,000	33	.6	94.0
10,000 TO 30,000	126	2.4	95.4
30,000 TO 60,000	63	1.1	96.5
60,000 TO 120,000	40	.8	97.3
120,000 TO 180,000	34	.7	98.0
180,000 TO 300,000	16	.3	98.3
MORE THAN 300,000	35	.7	100.0
TOTAL	5064	100.0	

TABLE 15B

AMOUNT OF CORE USED

CATEGORY LEVEL	FREQUENCY	PERCENT	CUM (PCT)
LESS THAN 10,000	3583	68.7	68.7
10,000 TO 50,000	707	13.5	82.2
50,000 TO 100,000	284	5.4	87.6
100,000 TO 150,000	132	2.5	90.1
150,000 TO 200,000	88	1.7	91.8
200,000 TO 250,000	64	1.2	93.0
MORE THAN 250,000	361	7.0	100.0
TOTAL	5064	100.0	

PP USAGE by jobs presents a decreasing, skewed curve which diminishes geometrically with time. With just about 66.8 percent of the jobs using one or less minutes, the high end of the scale shows .9 percent using over 90 minutes. The CON-DESCRIPTIVE data (Table 6) indicates that a total of 25.03 PP minutes, or 1.25 PPs were in use daily:

$$((\#PPs * 60 \text{ Min} * 24 \text{ Hrs} * 24 \text{ Days}) / \text{Minutes used}) / \#PPs \text{ avaiable} = \text{Use} \qquad \text{(eqn 4)}$$

This low number, representing an average of just over 6 percent of the 20 available PPs, may possibly be explained by the Percent of Control Point Usage in Table 7. There, the 36.10 percent indicates a low multiprogramming rate. Hence the low use of the Peripheral Processors.

SWAP OUT TIME. This category, used to account for all non-processing time a job experiences (less time spent in wait queues), reveals an interesting phenomenon. Although 41.4 percent of the jobs sampled experienced no swap out time, 564 jobs (10.9 percent) were swapped out 30 or more minutes, 97 of which jobs remained swapped out in excess of 90 minutes. Swapped jobs, depending on the reason for being swapped, could tie up one of the available control points thus contributing to the degradation of multiprogramming.

CPU TIME, the category in which most cursory judgements are made concerning the number of additional jobs the computer can potentially handle, must be viewed in conjuction with the CPU utilization rate of Tables 6 and 7. Of available CPU time, the utilization rate for all jobs for the month was 46.22 percent while the average CPU time

TABLE 16A

PERIPHERAL PROCESSOR USAGE

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
UNKNOWN	46	.9	.9
LESS THAN 1 MINUTE	3487	66.8	67.7
1 TO 5 MINUTES	765	14.7	82.4
5 TO 10 MINUTES	319	6.1	88.5
10 TO 15 MINUTES	171	3.3	91.8
15 TO 20 MINUTES	132	2.5	94.3
20 TO 30 MINUTES	110	2.1	96.4
30 TO 60 MINUTES	101	1.9	98.3
60 TO 90 MINUTES	39	.8	99.1
MORE THAN 90 MINUTES	49	.9	100.0
TOTAL	5064	100.0	

TABLE 16B

TIME SWAPPED OUT

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
NONE	2160	41.4	41.4
LESS THAN 1 MINUTE	1047	20.1	61.5
1 TO 5 MINUTES	685	13.1	74.6
5 TO 10 MINUTES	286	5.5	80.1
10 TO 15 MINUTES	135	2.6	82.7
15 TO 20 MINUTES	80	1.5	84.2
20 TO 30 MINUTES	262	5.0	89.2
30 TO 60 MINUTES	384	7.4	96.6
60 TO 90 MINUTES	83	1.6	98.2
MORE THAN 90 MINUTES	97	1.8	100.0
TOTAL	5064	100.0	

for the sample jobs was just 2.82 minutes. Computing the utilization rate of the 249 jobs (4.8 percent) in Table 17A which used in excess of 20 minutes, it turns out that those jobs (an average 10.4 per day) used 32 percent of the CPU usage time.

a) TOTAL SHIFT HOURS (CONDESCRIPTIVE DATA) = 245

b) 95.2% OF SAMPLE JOB (TABLE 16A) HOURS = 172

thus

$$\text{CPU Utilization} = 1-(b/a) = 0.32 \qquad \text{(eqn 5)}$$

A further analysis of CPU utilization as it relates to THRUPUT, TURNAROUND and ARRIVAL is presented in hourly increments in Figure 8. Close scrutiny of the diagram reveals a definite turnaround bottleneck during the first part of a day. The CPU utilization per [illegible] is fairly consistent throughout the day even through rapid increases in job ARRIVAL around the 8:00 AM hour. Additional examination of the phenomenon is discussed under hypothesis testing.

CM TIME. This variable shows that nearly 84 percent of all jobs spent 5 or less minutes in Central Memory while 2 percent consumed more than an hour of CM time, 53 jobs of which are over two hours.

TABLE 17A

CENTRAL PROCESSOR TIME

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
UNKNOWN	8	.2	.2
LESS THAN .1 MINUTE	2708	51.9	52.1
.1 TO .2 MINUTES	298	5.7	57.8
.2 TO .4 MINUTES	356	6.8	66.6
.4 TO .6 MINUTES	170	3.3	69.9
.6 TO .8 MINUTES	116	2.2	72.1
.8 TO 1 MINUTE	69	1.3	73.4
1 TO 5 MINUTES	880	16.9	90.3
5 TO 10 MINUTES	211	4.0	92.3
10 TO 20 MINUTES	154	3.0	95.3
MORE THAN 20 MINUTES	249	4.7	100.0
TOTAL	5064	100.0	

TABLE 17B

TIME IN CENTRAL MEMORY

CATEGORY LABEL	FREQUENCY	PERCENT	CUM (PCT)
LESS THAN 1 MINUTE	3760	72.0	72.0
1 TO 5 MINUTES	620	11.9	83.9
5 TO 10 MINUTES	284	5.4	89.3
10 TO 20 MINUTES	223	4.3	93.6
20 TO 40 MINUTES	176	3.4	97.0
40 TO 60 MINUTES	51	1.0	98.0
60 TO 90 MINUTES	37	.7	98.7
90 TO 120 MINUTES	15	.3	99.0
MORE THAN 2 HOURS	53	1.0	100.0
TOTAL	5064	100.0	

BOTTLENECK HYPOTHESIS

A summary analysis of the statistical results of the workload variables can best be seen through the Kiviat (Ref:28) graph of Figure 9 and the accompanying utilization table, Table 18. The graph, a pictorial representation of the time spent by a job at key processing points, reveals that the overhead activities are the major bottlenecks in a job's turnaround time. POSTPROC, the activity which occurs after internal execution has completed, accounts for 25 percent of the overhead. INQUEUE, the time spent waiting for initial control point assignment, adds another 24 percent. SWAPOUT time, although a function of how many jobs are concurrently in the system, is nevertheless additional overhead and contributes 5 percent to the total. IOWAIT, a result of peripheral equipment not being ready to use while a job is at a control point, contributes another 10 percent. Thus, IOWAIT, together with POSTPROC, INQUEUE, and SWAPTIME, account for 64 percent of the time a job spends in the system.

In order to get a more positive picture of this probable bottleneck, two additional SPSS runs were made on the workload data. The first used the strata of jobs which had a turnaround of one hour or more. This was done using:

SELECT IF (TURNAROUND GE 60.00)

The second run used the strata of jobs which completed in one minute or less:

SELECT IF (TURNAROUND LE 1.00)

The results provide the first concrete hypothesis as to what causes job elongation in the system. Assessing the jobs with processing times greater than one hour (Table 19 and

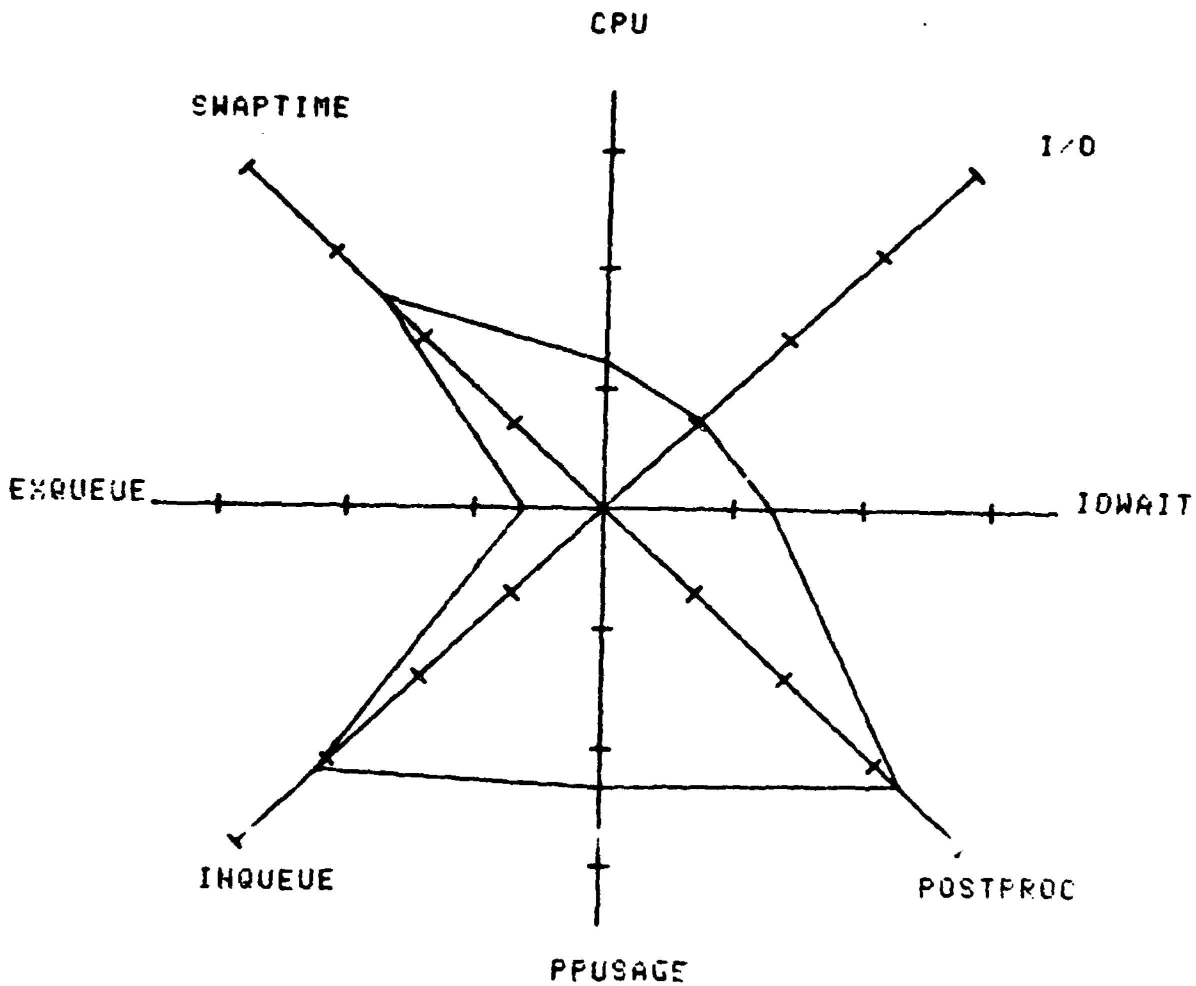


Fig 9. Relationship of processing activities to TURNAROUND - all jobs

Figure 10), the INQUEUE time increases by more than 43 percent from 24 to 34.4 percent while the remaining overhead variables - SWAPTIME, IOWAIT and POSTPROC - experience a significant decrease. In contrast, jobs which completed in one minute or less experienced the opposite (Figure 11). INQUEUE time dropped from 24 to 1.8 percent while SWAPTIME, IOWAIT and POSTPROC all increased on the average 50 percent. Counter to those changes is a surprising stability of the execution parameters, I/O, CPU and EXQUEUE. It turns out that regardless of the time a job spends in the machine, the percentage of time spent executing is a stable factor. Thus the hypothesis is established that the peripheral and time resources used by a job is the primary cause of job elongation.

TABLE 18

JOB TURNAROUND

RESOURCE UTILIZATION PERCENTAGES

VARIABLE	MEAN VALUE	PERCENT OF TURNAROUND
INQUEUE	6.25	24.0%
CPU	2.45	9.0%
I/O	2.09	8.0%
EXQUEUE	1.23	5.0%
SWAPTIME	4.93	19.0%
IOWAIT	2.58	10.0%
POSTPROC	6.53	25.0%
		100%

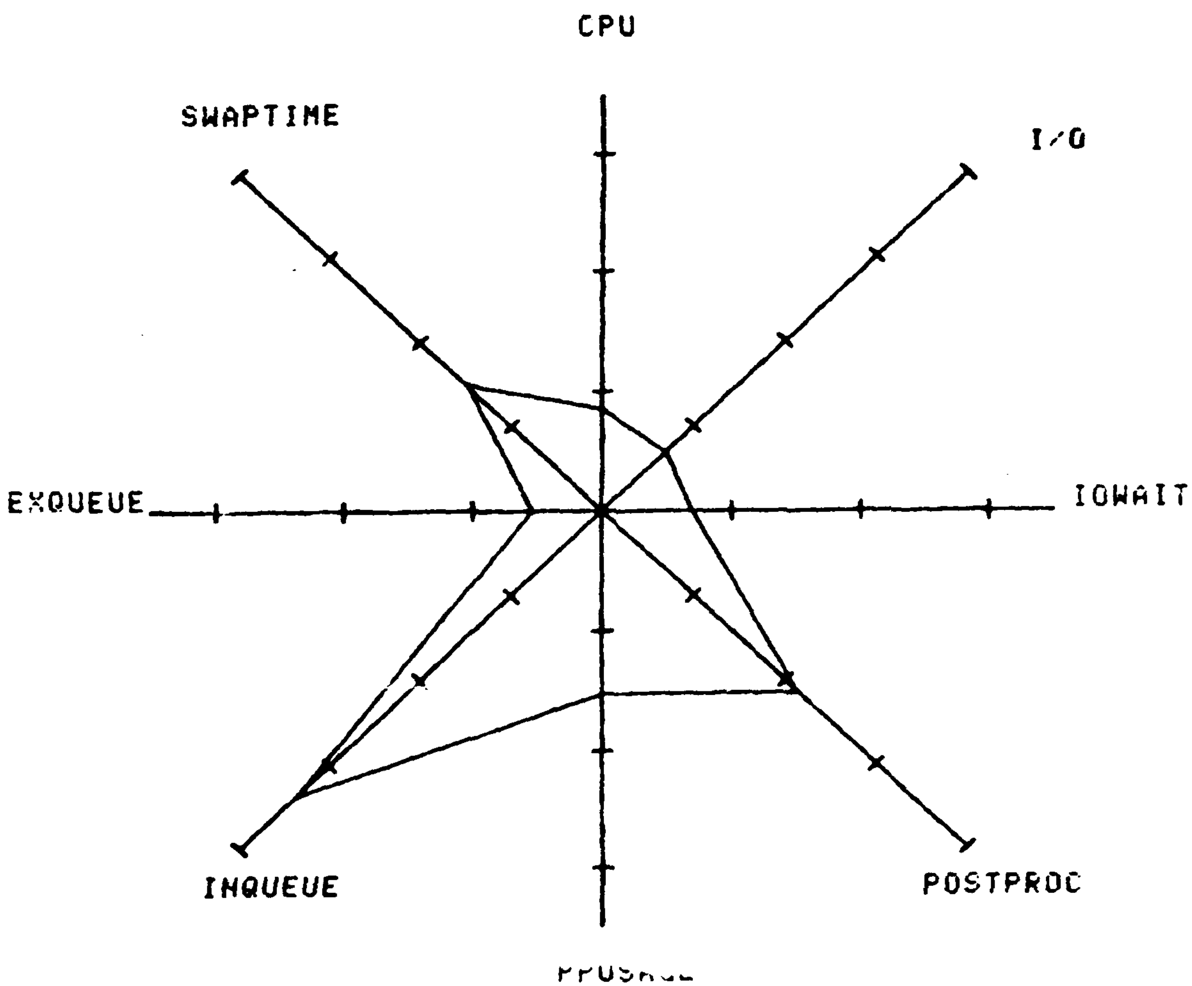


Fig 10. Relationship of processing activities for TURNAROUND > 60 min.

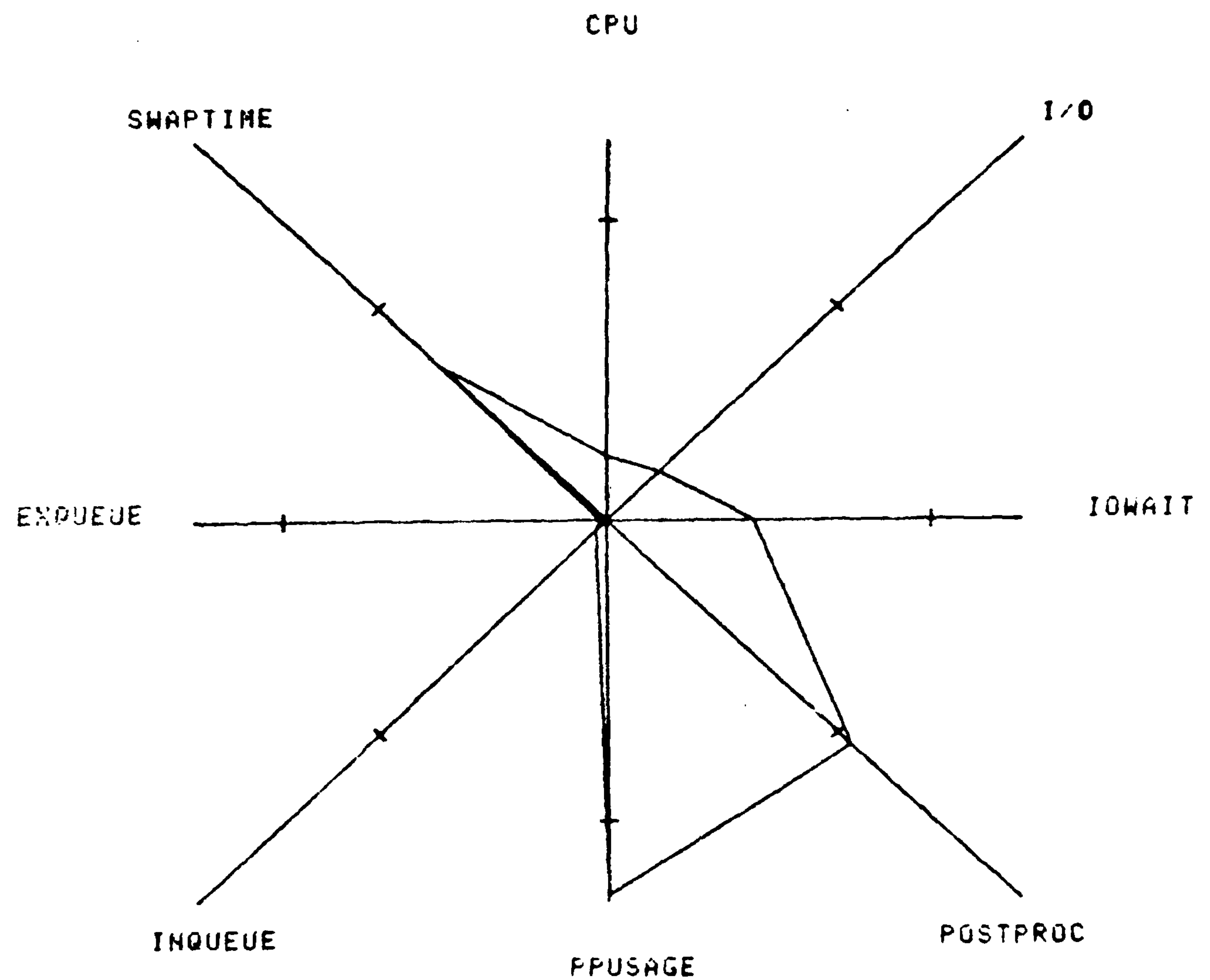


Fig 11. Relationship of processing activities for TURNAROUND < 1 minute

TABLE 19

RESOURCE UTILIZATION PERCENTAGES

FOR HIGH/LOW TURNAROUND TIME JOBS

	JOBS WITH TURNAROUND GREATER THAN 60 MIN		JOBS WITH TURNAROUND LESS THAN 1 MINUTE	
VARIABLE	MEAN VALUE	PERCENT	MEAN VALUE	PERCENT
INQUEUE	67.38	34.40	.010	1.81
POSTPROC	42.72	21.80	.212	38.48
IOWAIT	14.08	7.20	.090	16.33
SWAPTIME	29.72	15.10	.146	26.50
I/O	13.84	7.10	.046	8.35
CPU	17.05	8.88	.043	7.80
EXQUEUE	10.73	5.60	.004	.73
TOTALS	196.24	100.0	.551	100.0

SUMMARY

This chapter was used primarily to make raw observations about the workload data on which later analyses will be based. The observations will be expanded in the next chapter to make initial hypotheses about where major bottleneck areas can possibly be found. In subsequent chapters, extended analyses of the problem areas using the data will aid in building explanatory mathematical models as a baseline for predicting future thruput behavior under hypothetical changes.

V INITIAL HYPOTHESES

BACKGROUND.

The development of the initial hypotheses was aided considerably by theories offered by AFLC operators, schedulers and system programmers during interviews conducted in May, 1981. Those are included here without specific reference, along with several generalized hypotheses resulting from the analysis of the benchmark data statistics presented in the previous chapter. The hypotheses, where possible, have been organized around three improvement categories defined in RAND R-549-1-PR (Ref:5):

(1) Reducing the system workload

(2) Tuning the existing system

(3) Upgrading the computer system

REDUCING THE SYSTEM WORKLOAD.

HYPOTHESIS 1: A FREE-GOOD APPROACH HAS LED TO LARGE DEMANDS. Given a zero cost for computer resources and time, and the urgency placed upon conversions by the 7080 problem (Ref 5:14), design efficiency is not always of prime concern to programmers. AFLC's computer services are allocated to authorized users on a first-come/first-served basis and not on a cost basis (although resource usage costs, by organizational user, are produced and published (Ref 5:27; 31).

HYPOTHESIS 2: UNCONVERTED WORKLOAD HAS CAUSED INEFFICIENT SYSTEM USE. Critical time constraints on completing conversions from the 7080 to the CYBER may have forced programmers to use the new host computer as merely an extension of the old one. Techniques from the previous system may be extremely poor when applied to the newer equipment. Inappropriate use of I/O facilities, memory use and instruction mixes from a uni-programming machine may not effectively take advantage of the capabilities inherent in a multi-programming computer (Ref 5:28).

HYPOTHESIS 3: ORGANIZATIONAL METHODOLOGY DIFFERENCES COMPROMISE EFFICIENCY. An unfortunate difference in methodology between the Technical Support, the Users, and the Operations groups may exist which conflicts with a proven policy which provides optimal thruput for the computer. That is, arrangements that are reasonable for each group individually, may not meet with the objectives of the overall command viewpoint (Ref 5:31; LMT Interviews).

TUNING THE SYSTEM.

The structure of the computer may make its performance susceptible to a number of minor details about hardware, software, load and operating procedures. Tuning, the process of changing these details to make relative improvements in performance in specific areas can, when applied in sequence, lead to a considerable increase in overall performance.

HYPOTHESIS 4: APPARENT MINOR ACTIONS BY USERS ARE HAVING STRONG ADVERSE AFFECTS. If, for example, users have heavily I/O bound jobs, an increase in performance can be achieved by using larger blocking factors, and/or by using disk for intermediate job steps while using tape only for the initial input and final output (Ref 5:27; 1:12).

HYPOTHESIS 5: SCHEDULING BY SHIFT HAS COMPROMISED MULTIPROGRAMMING CAPABILITIES. External scheduling may tend to place only test jobs with heavy I/O activity in the prime shift leaving CPU-bound jobs in the off-prime shift. The result may be inefficient machine usage (Ref Figure 8C; 30).

HYPOTHESIS 6: INAPPROPRIATE STATIC INTERNAL SCHEDULING PRIORITIES COMPROMISE EFFICIENCY. General purpose internal dispatching priorities may give more effective CPU time to compute-bound jobs than to I/O jobs. If a higher percentage of the computer operation is I/O-bound then the I/O jobs should receive either dispatching priority or larger CPU time slices, or both, than the CPU-bound jobs (Ref 5:28).

HYPOTHESIS 7: I/O CONTENTION FOR A SPECIFIC DEVICE SLOWS PROCESSING. Concentration among different tasks for a Disk head, Controller, or Channel can cause severe inefficiencies on the machine (Ref 5:23).

HYPOTHESIS 8: INAPPROPRIATE OPERATING SYSTEM PLACEMENT REDUCES SYSTEM PERFORMANCE. Heavily accessed resident utility programs may be inefficiently contending for the same channel or resident areas when alternate channels or resident areas for compatible utility programs should be used to increase efficiency (Ref 37:135).

HYPOTHESIS 9: TOO FEW ACTIVITIES IN THE SYSTEM. Heavy CPU jobs may be wasting resources such as CM, tape drives, disk drives, etc., when run in a sparse job mix. A more balanced mix of jobs may improve overall machine thruput (see Table 19, Page 73).

HYPOTHESIS 10: SPECIAL REQUIREMENTS ARE WASTING PROCESSOR CAPABILITIES. Private device sets, tape mounting and card loading between jobs and job steps may accumulate to an hour or more over a day.

A different philosophy or hardware/software configuration can add valuable processing effectiveness to active computer time (Ref:1).

UPGRADING THE COMPUTER SYSTEM.

Increasing or upgrading computer resources can be a major factor in improving performance. The following hypotheses indicate problem areas that may provide insight to upgrading the computer system as a primary alternative solution.

HYPOTHESIS 11: INSUFFICIENT MEMORY IS CAUSING A BOTTLENECK IN THE CM QUEUE. A job may have started processing using a minimum memory allocation but is held up on subsequent steps which require larger memory blocks to process. Throughput rates can be improved if memory is augmented while keeping the scheduling algorithm static when excessive CM queue waiting is the result of insufficient memory (Ref 5:38).

HYPOTHESIS 12: INSUFFICIENT PERIPHERAL CAPABILITY CAUSES A BOTTLENECK. Increasing the limiting resource of the computer, such as tape drives, disk drives, printers or front-end computers, could vastly improve performance throughout a system. (See Figs 9-11)

HYPOTHESIS 13. A SERIES OF INTERNAL BOTTLENECKS ARE CAUSING THE INPUT QUEUE TO OVERLOAD. Based on the previous assumptions as to which areas

are causing bottlenecks at specific resource queues, the gross effect may prevent jobs from entering the system on a timely basis. (Figs 9-11, Pages 70-72)

HYPOTHESIS 14. THE WORKLOAD WILL EXCEED CAPACITY WITH THE ADDITION OF X SYSTEMS. Given the current workload (Ref:30) and the average utilization percentages, the anticipated workload (Figure 6, Page 40) will exceed the computer's capability to process all jobs with the addition of X number of systems. The current workload, consisting of 2088 hours of monthly processing, is increasing at a rate of 4.08 percent per month (see Production Jobs, Figure 6, Page 40). The current load represents 94 percent of capacity (2088/2225) with 138 hours (multiprogramming) remaining. The machine will overload when the remaining 138 hours are exceeded.

VERIFICATION REQUIREMENTS

HYPOTHESIS 1. A Free-Good Approach Has Led to Large Demands. This hypothesis can be tested by examining the DAYFILE accounting data for blocking factors used by jobs. Files written to tape by the user system is most efficient when a full 512 CM words (5120 characters) are packed into a block of data. If it is found that jobs are using smaller blocking sizes, e.g., 3000 or less as is used for the 7080 Computer, more inter-record gaps and more I/O requests will be required resulting in higher system overhead and the use of more tapes. As an example, if the average blocking for all tape jobs is 3000 characters, increasing the blocking to the tape maximum of 5120 would decrease I/O overhead by 41 percent.

HYPOTHESIS 2. Unconverted Workload Has Caused Inefficient System Use. Testing the validity of this hypothesis requires simulation or emulation of the 7080 computer on the CYBER and a detailed look into the programming techniques being used

by programmers of the converting systems. Considering the conversion schedule (Figure 6, Page 40), the period for which this phenomenon is expected to persist (one and a half years) in relation to the probable payoff (less than a year and a half of increased machine time), the time and cost required for the study is not warranted.

HYPOTHESIS 3. Organizational Methodology Differences Compromise Efficiency. Since each of the organizations (user, operations, and technical support) are co-equal and most likely using methods which best accomplish their specific goals within constraints of personnel and time, this hypothesis must be viewed in terms of the larger organizational objective. That is, the end product output requirement.

(1) Activities by operators can be examined through observation of the DAYFILE data as they apply to operator initiated action. The SWAP rate (Table 16B, Page 66) as well as the number of lockins/lockouts per day, can be a clue to this activity.

(2) Scheduling bottlenecks, as observed by the ARRIVAL rate (Table 9A, Page 53) could be an indication that the user's timing requirements (output, interleaving) may be unreasonable or out of line with the most efficient daily thruput processing.

(3) An examination of job thruput as it relates to job priorities could indicate that a different priority scheme may have to be initiated by the technical group.

Further examinations of the DAYFILE to test (1) above would require the development of an additional DAYFILE scanning program with no certainty that the payoff would warrant the time and cost involved. Point 2 above can easily be tested by reassigning some jobs to lighter scheduled periods and observing the results. But a considerable amount of time would then be necessary to negotiate with users to permit

permanent rescheduling. The testing of different priority schemes would require software personnel to develop and test several combinations without assurances that any of the schemes would payoff.

<u>HYPOTHESIS 4</u>. Apparent Minor Actions by Users are Having Strong Adverse Affects. This hypothesis can be tested by obtaining information about workload characteristics from users regarding how and why certain files are built, stored and used in their present form. The payoff could result in an I/O overhead saving similar to that of hypothesis 1.

<u>HYPOTHESIS 5</u>. Scheduling by Shift Has Compromised Multiprogramming Capabilities. This can be checked by looking at the CPU and I/O activity per job during the relevant periods. If the test proves positive (i.e., a change in when jobs are processed produces more multiprogramming throughout the day) implementation of the new procedure may be at the expense of user testing/development and a possible delay in expected implementation of future workloads.

<u>HYPOTHESIS 6</u>. Inappropriate Static Internal Scheduling Priorities Compromise Efficiency. Testing whether this phenomenon exists is accomplished by determining the ratio of I/O bound jobs to Compute bound jobs, and the relative weight given each by the scheduling algorithm. The payoff, should the test prove a different scheme is more efficient, would be a reduction in thruput and, hence turnaround.

<u>HYPOTHESIS 7</u>. I/O Contention For A Specific Device Slows Processing. A determination as to whether a test for this hypothesis is necessary can be made through an observation of two Table 8 factors, DISKIO and DISKWAIT. A calculation of the ratio of I/O service (DISKWAIT) to unit record processing (DISKIO) would produce a

measurable statistic for which to test the hypothesis. A ratio greater than .04 (this is the effective transfer rate per disk rotation given constant access to disk and channel) would confirm the initial theory and warrant further testing. The additional testing involves actually changing the location of a sampling of files and running a test job stream to verify changes in performance.

HYPOTHESIS 8. Inappropriate Operating System Placement Reduces System Performance. This hypothesis can be tested by experimenting with different combinations of utility program placement. As noted in Chapter III on Software Organization, multiple access utilities are stored in a common memory area for use by all jobs. While a duplicate placement of such utilities would increase core usage, its resultant increase in thruput may well be worth the trade-off.

HYPOTHESIS 9. Too Few Activities In The System. This hypothesis, derived from observing a high TURNAROUND vis-a-vis job ARRIVAL during the 0000-0800 period (Figure 8A-C, Pages 50-52) as opposed to a relative low TURNAROUND with a higher ARRIVAL rate during the 1600-2400 period, can be tested by experimenting with different job mixes during those periods. Should the shifting of jobs to different periods result in an increase in overall turnaround, the permanent change would be warranted.

HYPOTHESIS 10. Special Requirements Are Wasting Processor Capabilities. As noted under Peripheral Devices (Chapter III) the system uses 24 of its 33 disk drives for private device sets. These devices are non-allocatable and, as such, can only be assigned to one job at a time. Thus, operators must mount and dismount different sets as each new user system enters the computer and requests these devices. The hypothesis can be tested by accounting for all DISKWAIT time experienced by the sample jobs. The expected DISKWAIT time should be 0 for a balanced computer.

HYPOTHESIS 11. Insufficient Memory is Causing a Bottleneck in the CM Queue. Testing this hypothesis involves the use of a hardware monitor to gather, average and compare memory wait times against memory usage (Table 17B, Page 67). The criteria for determining whether such a test should be made is derived from an observation that memory usage is a major factor in THRUPUT time as given by a regression analysis of THRUPUT against CM USAGE. If the regression warrants the monitor test and the results compared produces a large ratio (i.e., memory wait to memory use), memory is probably an important constraint and should be augmented.

HYPOTHESIS 12. Insufficient Peripheral Capability Causes a Bottleneck. This hypothesis is a derivation from an initial examination of postprocessing requirements and the difference between THRUPUT and TURNAROUND. The hypothesis can be tested by temporarily adding a printing device to see if the bottleneck can be eliminated. This test may prove that a printer is the limiting resource or it may indicate that disk head contention would cancel any beneficial effects.

HYPOTHESIS 13. A series of Internal Bottlenecks are Causing the Input Queue to Overload. This hypothesis can be verified through the use of an input queue model which will indicate the expected time a job will spend in the input queue. If the observed time exceeds the expected time the hypothesis is considered verified.

HYPOTHESIS 14, Workload Will Exceed Capacity With The Addition of X Systems. This hypothesis can be evaluated by assessing the average hours consumed by current systems and extrapolating a sum of the means to a point at which the machine capacity is reached. The number of times the mean is added to the sum beyond the current sum of the system means would then represent the number of average systems which can be added.

VI DEVELOPMENT OF A TURNAROUND TIME MODEL

OVERVIEW

The primary analysis tool used in this thesis is an analytic model incorporating multilinear regression analysis and factor analysis techniques to theorize the causes of the thruput problem. This technique, as explained in Chapter II, provides first an explanatory vehicle to get answers as to why the expected performance of the computer has not been achieved. Secondly, it provides a proven predictive tool to assess the impact of changes in the system's workload and internal components (Ref:12). And thirdly, a method for verification as to whether the changes actually improved performance is possible with the model.

Previous studies, accomplished by Hartrum (Ref:23) and Thompson (Ref:36), provided an excellent starting point for the model development. Several key parameters which were not obtainable at the time have since been derived from the reduction programs (see Chapter II) and are used to enhance those applications.

MODEL STRUCTURE

Basically the analytic model is developed around an organized description of the job flow components of the CYBER (see Figure 9, Page 70). The model identifies potential points where activities may be delayed as a job processes through the system.

As a rule, turnaround time would include all activities involved in job processing from the user to the user (i.e., submission, staging, processing, routing and receipt). The model developed for this thesis, however, considers only the processing phase. Therefore, turnaround time, for the purpose of this investigation, is defined as the time from job input to the system (ARRIVAL) through job output (Final Print). Thruput then, is the time required for internal processing which excludes the time from ARRIVAL to the time the job is assigned to a control point, and from control point exit through final print.

Figure 12 depicts the complete TURNAROUND Time Model. Figure 13 shows the model in tree form, and Table 20 contains the model variables. The model itself is developed incrementally using testable components to lead to the final model. The basic model is defined as:

TURNAROUND = QUETIME + THRUPUT

where

THRUPUT = EXECUTION + WAITTIME

QUETIME = INQUEUE + POSTPROC

and

EXECUTION = CPU + IO + EXQUEUE

WAITTIME = SWAPTIME + DISKWAIT + TAPEWAIT

These dependent variables will be modeled as a function of the independent variables to determine which of the independent variables caused the most delay. First,

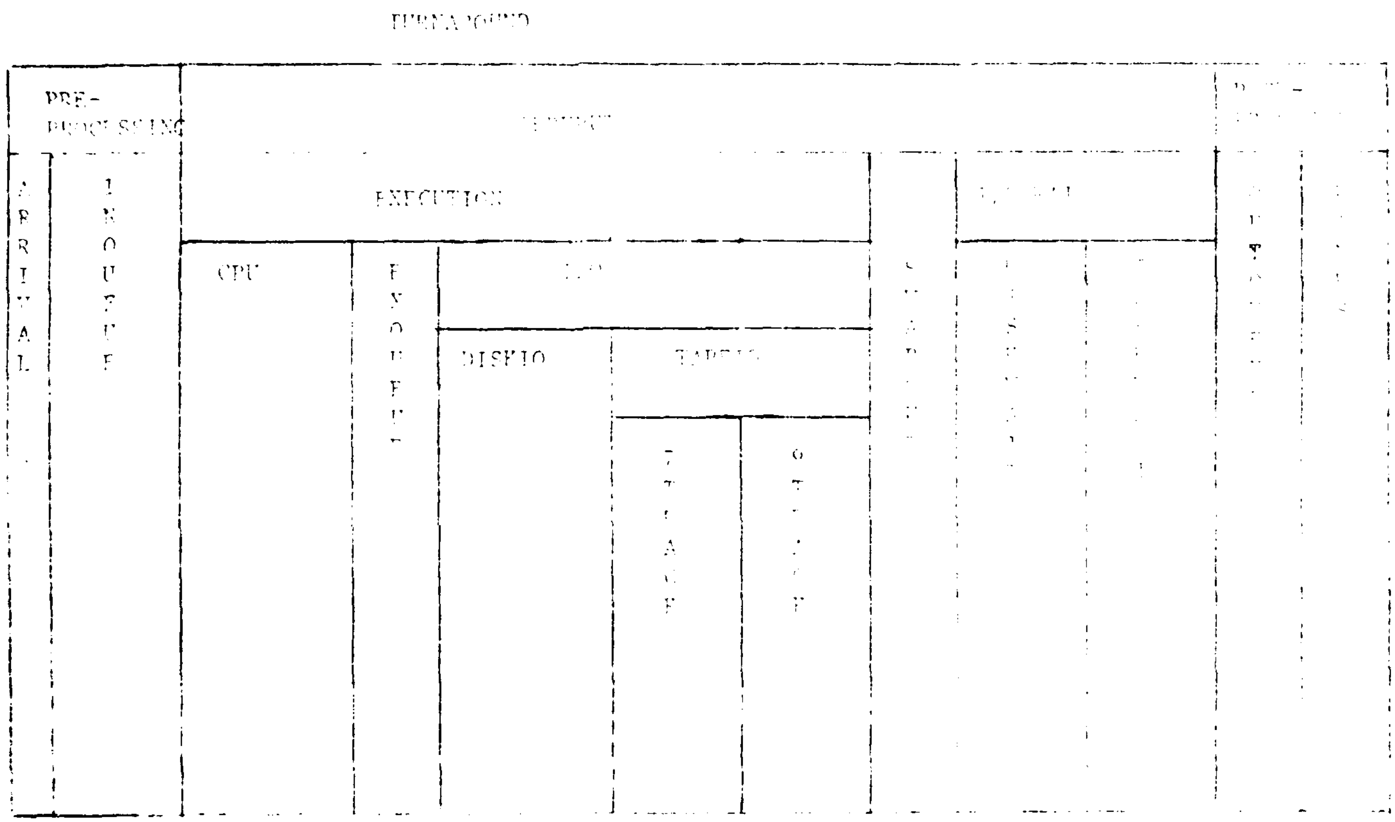


Fig 12. Turnaround Time Model

AD-A115 578 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
THRUPUT ANALYSIS OF AFLC CYBER 73 COMPUTERS.(U)
DEC 81 L T BONDURANT
UNCLASSIFIED AFIT/GCS/EE/81D-2 NL

2 of 2

END
DATE
FILMED
7 82
DTIC

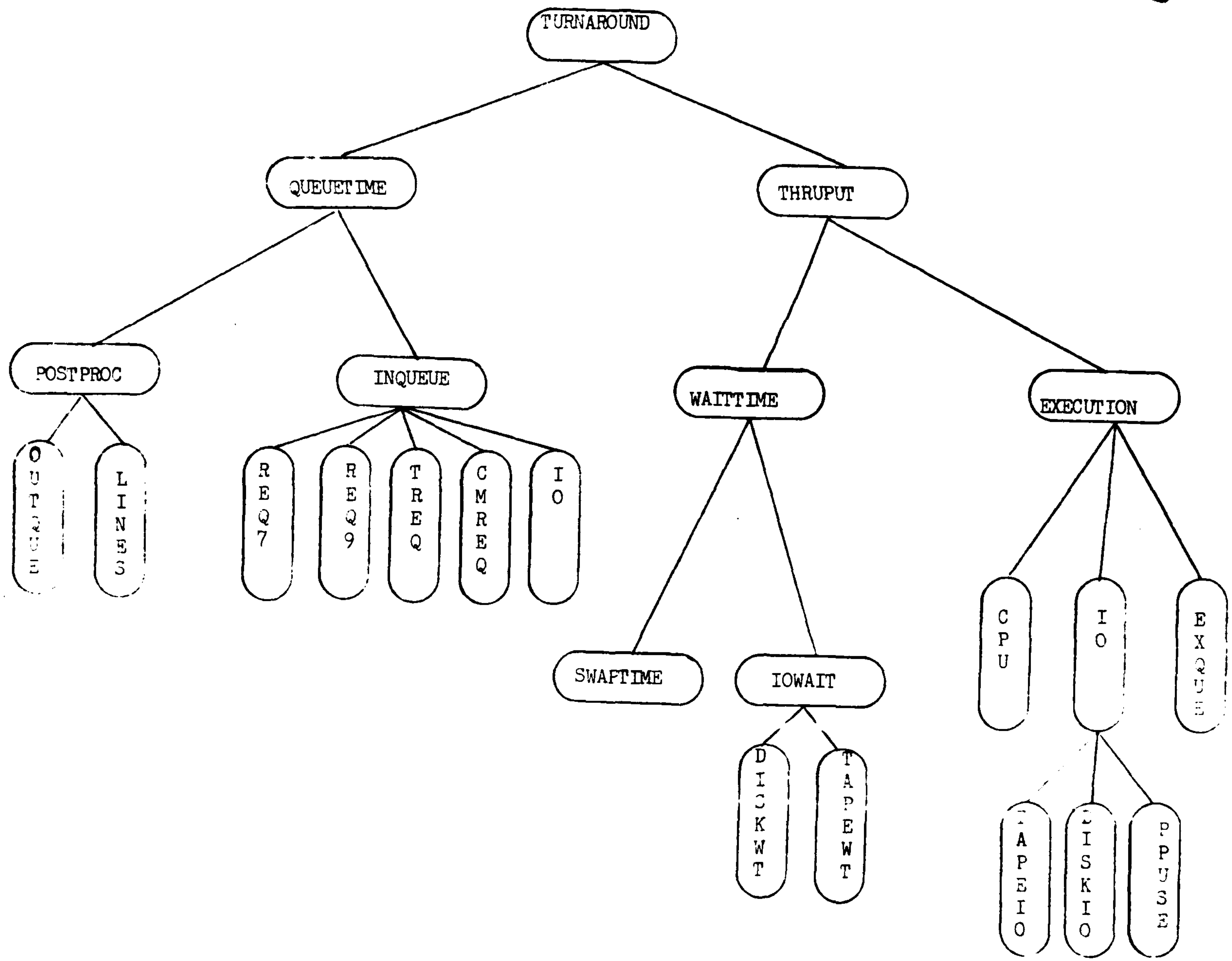


Fig 13. Turnaround Model Tree Structure

TABLE 20

VARIABLES FOR THE

TURNAROUND TIME MODEL

VARIABLE	DESCRIPTION
TURNAROUND	TURNAROUND TIME
THRUPUT	THRUPUT TIME
INQUEUE	TIME IN INPUT QUEUE
IO	I/O TIME
EXECUTN	TIME SPENT EXECUTING
SWAPTIME	TIME SPENT SWAPPED OUT
WAITTIME	TIME SPENT WAITING I/O RESOURCES
POSTPROC	POST PROCESSING TIME
OUTQUEUE	TIME SPENT IN THE OUTPUT QUEUE
ARRIVAL	TIME OF JOB ARRIVAL
EXQUEUE	TIME SPENT IN EXECUTION QUEUES
CPU	CPU TIME
CM	TIME SPENT IN CENTRAL MEMORY
DISKWAIT	TIME SPENT WAITING FOR DISK
TAPEWAIT	TIME SPENT WAITING FOR TAPE
DISKIO	NUMBER OF DISK ACCESSES
TAPEIO	NUMBER OF TAPE I/Os
NINETRK	NUMBER OF 9 TRACK TAPES USED
CORE	AMOUNT OF CORE USED
MAXMEM	MAXIMUM MEMORY USED
LINES	LINES PRINTED

however, THRUPUT and QUETIME are modeled separately to detect any existence of multicolinearity or non-additivity between the independent variables before using the two as independents of TURNAROUND.

QUETIME MODEL. The Preprocessing and Postprocessing aspects of a job, to one degree or another, should depend on the resource requirements for the job. Post-processing, however, is likely to be independent of the resource parameters governing preprocessing since printing is the sole remaining task for the job once internal processing is complete. Inqueue, however, is modeled separately in an attempt to determine the source variation among the component parameters before testing their goodness as predictors for overall QUETIME.

INQUEUE. The expected time a job will spend in the input queue in an unsaturated computer should depend on the resources requested and the resources available when the request is made. The INQUEUE time then, is a function of the following parameters which the user enters on his job card.

TABLE 21

INQUEUE MODEL VARIABLES

INPUT PARAMETER	VARIABLE USED
- CPUTIME requested	= TREQ
- IOTIME requested (unavailable)	= actual IOTIME
- Central Memory requested	= CMREQ
- 7 Track Tapes requested	= REQ7
- 9 Track Tapes requested	= REQ9
- aging time already spent in the queue	= unknown

IOTIME requested was unavailable as can be seen but a reasonable approximation can be made by using the actual time observed. It should also be noted that in many cases the user will have estimated a much larger value than required for his job to ensure that his job will run to completion (the system will abort a job if actual resource use exceeds the requested time or an installation default time in cases where a request field is left blank). Thus the time in the input queue would depend on a large request value that might correspond to a small value actually used by the job. The INQUEUE model thus becomes:

$$\text{INQUEUE} = \text{A1} + \text{B1(TREQ)} + \text{B2(IO)} + \text{B3(CMREQ)} + \text{B4 (REQ7)} + \text{B5 (REQ9)} \qquad \text{(eqn 6)}$$

QUETIME then, is modeled as:

$$\text{QUETIME} = \text{INQUEUE} + \text{POSTPROC}$$

THRUPUT MODEL. Somewhat similar to QUETIME, the thruput model can be viewed as a function of the resources used while a job is at a control point. The times, however, include the time which the job spent in the CM queue due to being swapped or rolled out. This value is later subtracted from the control point time and modeled separately but the number of swaps/rolls and length of each is unavailable. The THRUPUT model then, is a function of:

- CPU time actually used
- PPUSAGE time actually used
- IO time actually used (will overlap with PPUSAGE)
- CM memory actually used

- DISKIO actually used
- TAPEIO actually used
- IOWAIT actual DISKWAIT + TAPEWAIT
- SWAPTIME time swapped out or pre-empted
- EXQUEUE time in CPU and I/O queues

It should be noted that a certain amount of interaction is expected between PPUSAGE, IO, DISKIO and TAPEIO. This interaction is a function of the number of I/O calls made by a job and not of the number of jobs concurrently in the system. Thus the model for THRUPUT should be:

$$\begin{aligned} \text{THRUPUT} = C1 &+ D1(\text{CPU}) + D2(\text{CM}) + D3(\text{PPUSAGE}) + \\ &D4(\text{IO}) + D5(\text{DISKIO}) + D6(\text{TAPEIO}) + \\ &D7(\text{EXQUEUE}) + D8(\text{IOWAIT}) + D9(\text{SWAPTIME} \qquad \text{(eqn 7)} \end{aligned}$$

EXECUTION MODEL. The time a job spends executing in the system directly affects the thruput time of a job and is comprised of CPU, IO and the time spent in the CPU and IO queues. The CPU and IO times, as noted before can be directly extracted from the DAYFILE statistics. Queue time for the model, although not available as an independent statistic of the DAYFILE data, can be derived from the CPU, CM and I/O statistics. According to the job flow algorithm (Chapter III) the CM time recorded is the time a job maintains its field length while performing CPU and I/O activities. Therefore, the difference between the two and CM would account for the queue time.

EXQUEUE = CM - (CPU + IO)

The execution model then, is a function of:

- CPU time spent in the CPU
- CM time spent in central memory
- IO time performing I/O operations
- EXQUEUE time waiting in execution queues

While the model becomes:

EXECUTION = E1 + F1(CPU) + F2(CM) + F3(IO) + F4(EXQUEUE) (eqn 8)

The basic TURNAROUND model then, as a function of the previously described models, can now be portrayed as:

TURNAROUND = G1 + H1(IO) + H2(POSTPROC) + H3(CPU) +
H4(IOWAIT) + H5(SWAPTIME) + H6(INQUEUE) +
H7(CM) + H8(PPUSAGE) + H9(DISKIO) +
H10(TAPEIO) + H11(EXQUEUE) (eqn 9)

MODEL TUNING

The results of the hypothetical models were tested through regression analysis procedures to verify their goodness in explaining the variation in turnaround time. The three dependent variables (TURNAROUND, QUETIME and THRUPUT) were regressed against the set of independent variables in a stepwise mode in order to find the optimal list of variables to be included in the final equation. The r^2 value was used as a criteria

in evaluating the models. Once the functional variables were determined an additional regression analysis was made regressing the dependent variable of each model against its respective set of independent variables to find and eliminate those which did not contribute to their model or which exhibited traits of multicolinearity or non-additivity. Table 22 shows the final results and the critical r^2 values obtained from the tests.

The r^2 results tend to show that the selected independent variables, with the exclusion of the job card variables (CMREQ, TREQ, REQ7 and REQ9) which explain very little of the variation in INQUEUE time, are adequate predictors for the turnaround models. Four of the variables originally in the THRUPUT model (DISKIO, TAPEIO, PPUSAGE and EXQUEUE) were found to have a high degree of multicolinearity with other variables of that group on an independent regression run of the THRUPUT model and subsequently discarded from the set.

After assessing the regression results, the CM variable, a weighted total of CPU and IO, was dropped in favor of CPU and IO, while INQUEUE was used as a surrogate for the request variables. The final model for TURNAROUND is given in equation 10 while Figure 14 shows the independent variables in a tree structure after the leaf nodes have been eliminated.

$$\text{TURNAROUND} = .14 + 1.24(\text{IO}) + 1.0(\text{POSTPROC}) + 1.21(\text{CPU}) + .56(\text{IOWAIT}) + 1.01(\text{SWAPTIME}) + .99(\text{INQUEUE}) \quad \text{(eqn 10)}$$

VALIDATION

The mean values for the independent variables collected from the SPSS runs (Table 6, Page 43) were used to drive the model and calculate the associated values of the final model. Equation 11 shows the results of the calculations.

TABLE 22

TURNAROUND MODEL REGRESSION ANALYSIS

DEPENDENT VARIABLE	INDEPENDENT VARIABLE	CRITICAL r^2 VALUE	CUM PERCENT OF VARIATION
INQUEUE	TREQ	.0098	.0098
	IO	.0032	.0131
	REQ7	.0016	.0146
	REQ9	.0013	.0160
	CMREQ	.0001	.0161
			.0161
THRUPUT	SWAPTIME	.6389	.6389
	CM	.2636	.9025
	IOWAIT	.0449	.9473
	IO	.0123	.9597
	CPU	.0141	.9738
			.9738
TURNAROUND	INQUEUE	.6091	.6091
	SWAPTIME	.1579	.7670
	POSTPROC	.1452	.9122
	CPU	.0617	.9739
	IO	.0126	.9865
	IOWAIT	.0071	.9936
			.9936

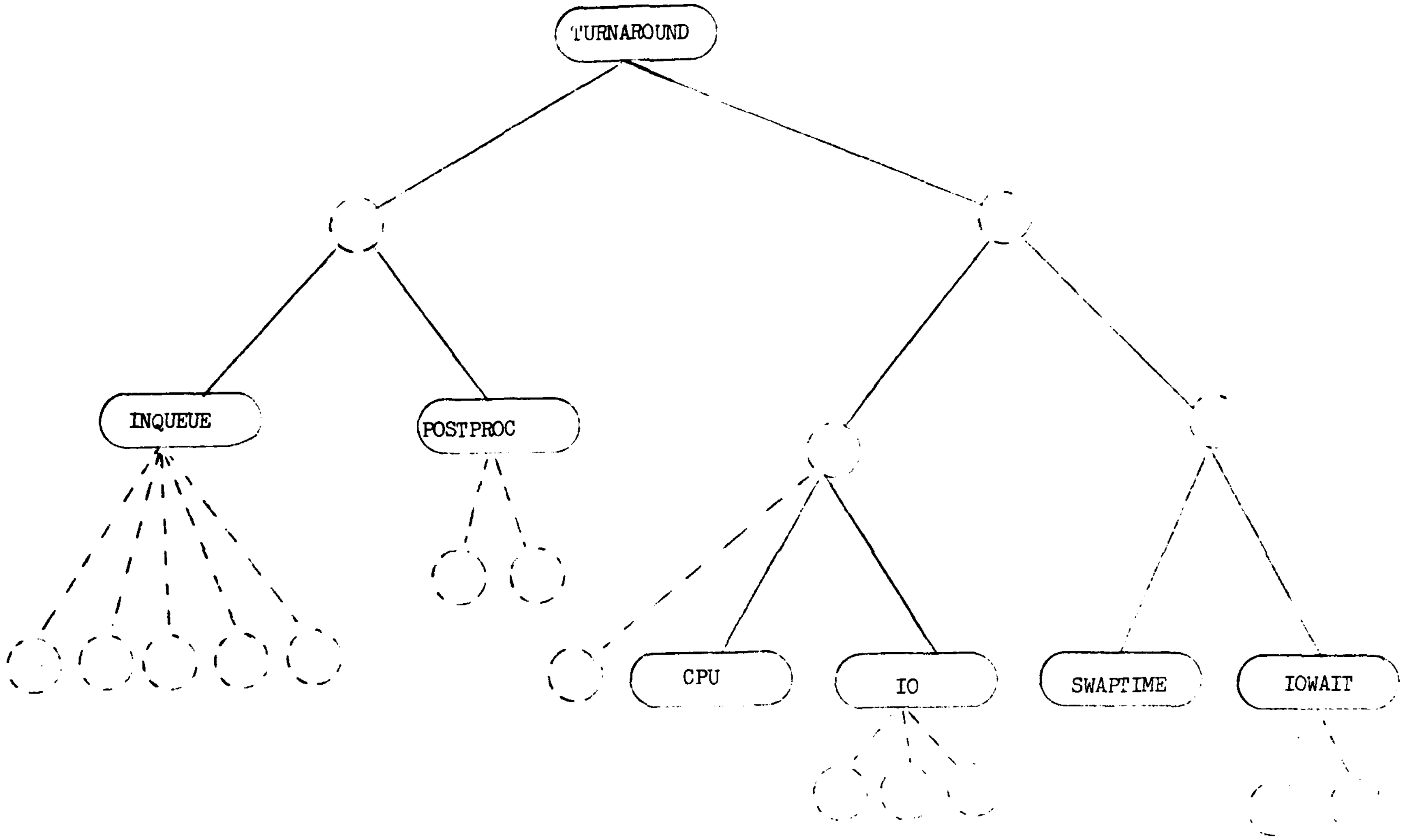


Fig 14. Tree Form of Final Turnaround Model

$$\text{TURNAROUND} = .14 + 1.24(2.09) + 1.0(6.53) + 1.21(2.45) + .56(2.58) + 1.01(4.93) + .99(6.25) = 24.84 \quad \text{(eqn 11)}$$

The validation indicates that the final model provides an excellent predictor for the true TURNAROUND since:

Actual TURNAROUND time = 24.74

Predicted TURNAROUND = 24.84

To obtain further assurances of the model's accuracy, a random sample (of 1 percent) of the jobs was selected from the sample population using

SAMPLE 0.01

on an SPSS procedure card. The mean values were derived in the usual manner with the following results

VARIABLE	MEAN	VARIABLE	MEAN
CPU	1.407	POSTPROC	12.548
IO	1.243	INQUEUE	11.157
IOWAIT	1.245	TURNAROUND	31.228
SWAPTIME	3.424		

thus

Actual TURNAROUND = 31.23

$$\text{Predicted TURNAROUND} = .14 + 1.24(1.24) + 1.0(12.55) + 1.21(1.41) + .56(1.25) + 1.01(3.42) + .99(11.16) = 31.14$$

SUMMARY

In this chapter the analyses of the workload observations (Chapter IV) were used as the basis for developing a deterministic model of a job's turnaround time in the computer system. The independent variables of the turnaround time were selected from the set of all variables extracted or derived from the benchmark workload. These were initially defined as the time spent waiting (Quetime) and the time spent processing (thruput). Models for those independent variables were then developed and tested to find, for each, a more definitive subset of variables which could be used as surrogates for QUETIME and THRUPUT in the final predictive TURNAROUND time model.

As would be expected in a search for an optimal model, numerous dead-ends were explored before finding the best combination of variables which would explain the actual turnaround time observed. The final model (IO, POSTPROC, CPU, IOWAIT, SWAPTIME and INQUEUE) provides an excellent tool for explaining the delay points in the computer system.

The coefficients of two variables, POSTPROC and INQUEUE, show that each unit change in the variable causes a corresponding unit change in turnaround. This can be expected since the two are primarily functions of how many jobs are in the system and not of the job itself. An indepth analysis of the variation in INQUEUE and POSTPROC will be made in the next chapter on hypotheses verification. IOWAIT, showing a coefficient of .56, is again a variable independent of a job's execution and will also be examined further in the next chapter. In general, however, the value of IOWAIT is purely a result of unavailable resources (see Rollout, Chapter III) and should represent a one-to-one relationship with turnaround (i.e., a unit increase/decrease in IOWAIT should cause a like change in TURNAROUND. One possible explanation is that less than half of

all jobs sampled experienced wait time for resources while the model averages the data points for the entire sample population. In fact, the coefficients of CPU and IO which are functions of thruput as is IOWAIT, appear to make up the difference in the IOWAIT statistic when the regression analysis treats all jobs as having values available for each variable. Suffice it to say that for the purpose of this thesis, given the available data, the final model accomplishes the task of explaining job turnaround time and will serve as a useful predictor of future job turnaround when the selected independent variables are varied according to hypothesized modifications.

VII HYPOTHESES VERIFICATION

INTRODUCTION

As mentioned in the section on requirements for hypotheses verification (Chapter V), several of the hypothesized problems required analysis procedures beyond the stated scope of the thesis and of the time available to conduct such tests. These are nonetheless analyzed and discussed, and while many of them are subsequently deferred to other possible efforts, others have been helpful in adding to the overall improvement of the problem through analytical deduction. The completely discarded hypotheses and the reasons for rejection are also included in the analysis which follows. A follow-on thesis effort should be considered in order to fully optimize to the machine capability.

RETAINED HYPOTHESES

HYPOTHESIS 1. A Free-good Approach Has Led to Large Demands. This hypothesis boils down to assumptions about the kind of blocking programmers are using for I/O operations, and, although an actual investigation was not made because of time constraints, should be included as a potential for time reduction. If it can be assumed that the hypothesis is valid, then the difference between the maximum blocking of the IBM 7080 and the CYBER amounts to a 41 percent decrease in I/O overhead:

3000 IBM blocking / 5120 CYBER blocking = 41 %

This reduces the effective average I/O time to

I/O = 2.09 - (2.09 * .41) = 1.23 min/job

HYPOTHESIS 3. Organizational Methodology Differences Compromise Efficiency. All aspects of the established testing procedures for this hypothesis cannot be accomplished under the scope of this thesis. One key test for example, a program to

detect the number of operator initiated lockins/lockouts, could be developed at a later time to find the exact occurances of that activity. A scheduling bottleneck as hypothesized however, does appear to exist in the current data and can be examined without additional programs. Figure 8A indicates a bottleneck during the 0800-1200 period. The exact reason for the sudden surge (the number of arrivals triples during the period - Table 9A) would have to be discovered by talking to the user organizations involved. A possible solution, apart from talking to the users, is addressed along with schedule changes under hypothesis 6.

HYPOTHESIS 5. Scheduling by Shift Has Compromised Multiprogramming Capabilities. This hypothesis, as with the last assumption, can only be partially tested with the current data. This hypothesis, pertaining to scheduling activities by shift was tested by clustering the jobs of the sample data according to the three work shifts of the data center. The mean CPU and I/O values were then derived and tested through a T-Test procedure to see if a difference existed by shift. A null hypothesis (Ho) was established that no difference existed between the CPU and I/O resource usage among the shifts. The alternative (Ha), should the null be rejected, would establish that a difference did indeed exist and warrant further investigation into the possibility of spreading the jobs more evenly over the day. The following table shows the test results at the .03 (1.96) Significance level:

TABLE 23

TEST OF DIFFERENCES BETWEEN WORK-SHIFT PROCESSING

SHIFT	VARIABLE	MEAN	T-VALUE	RESULT
0000-0800	CPU	3.093	2.79	REJECT Ho
	I/O	3.185	4.46	REJECT Ho
0800-1600	CPU	2.178	2.01	REJECT Ho
	I/O	1.634	3.74	REJECT Ho
1500-2400	CPU	2.313	- 0.42	ACCEPT Ho
	I/O	1.837	- 1.04	ACCEPT Ho

The rejection of the NULL establishes the proposition that some difference exists in CPU and I/O resources used by shift. Further testing then, through a special running of the CLARA program to extract the multi-programming level during the specific periods is necessary to completely verify the assumption.

HYPOTHESIS 6. References the internal scheduling algorithm. The premise here is that if a majority of high turnaround jobs are found to be I/O bound, then the priority scheduler should be changed to give a larger processing time slice to I/O type jobs than it gives to CPU bound jobs. An exact measurement of heavy I/O bound versus high CPU bound jobs can only be made with a hardware monitor which is beyond the purview of this thesis. Nevertheless, a reasonable surrogate for the test can be obtained from the DAYFILE data already extracted. Thus, an SPSS run to examine the THRUPUT times for jobs which used no tape facilities versus those that did produced the following results:

THRUPUT (tape only jobs)	=	35.48
THRUPUT (non tape jobs)	=	5.00
THRUPUT (all jobs)	=	11.97

The results clearly show that jobs using tape facilities tend to spend a longer period for thruput (recall that thruput does not include the input queue time or postprocessing time) than other jobs. Therefore, the hypothesis is proven warranting a change in process time allocation in favor of tape use jobs.

HYPOTHESIS 7. I/O Contention For Specific Device Shows Processing. This hypothesis, as detailed in Chapter V, can be tested through the development and execution of a Disk Service Model. Using the device characteristics (see Chapter III) the service time can be derived as:

I/O SERVICE TIME = Access Time + Rotational Delay + Block Transfer Time (eqn 12)

where

Block Transfer Time = 1 / (Trans Rate/Avg. Blocking)

= 1/ (1,130,000/640) = .00057 sec.

and

I/O Service Time = .040 +.0083 + .00057 = .04887 sec.

then

DISKIO:DISKWAIT = 1.31 / (.04887 * 7000 * 60) = .230

This exceeds the threshold of .04 established as the test criterion. Therefore, the hypothesis that I/O contention is a problem is proven and warrants further testing by changing the location of the user files to obtain a satisfactory ratio of DISKIO vs DISKWAIT.

HYPOTHESIS 10. Pertains to Special User Requirements such as private disk packs. Assessing the DISKWAIT time experienced by the sample jobs (Table 10A, page 55), 6,768 minutes or 112 hours were spent in July by jobs waiting for disks. This value is included as part of the IOWAIT statistic in the TURNAROUND model and can be extracted to show the expected decrease in TURNAROUND as a result of jobs not having to wait for the mounting of their private packs:

where

IOWAIT = TAPEWAIT + DISKWAIT = 2.58 min/job

AND

IOWAIT = IOWAIT - DISKWAIT

giving

IOWAIT = 2.58 - 1.31 = 1.27 min/job

HYPOTHESIS 12. Insufficient Peripheral Capability Causes a Bottleneck. This hypothesis can be initially tested with the use of a Printer Service Model and postprocessing time observed. As mentioned earlier, the postprocessing done by a job is entirely a function of its print requirements. Variation in this time variable beyond printing would depend on the number of jobs competing for the printers available to the computer which can be modeled as a queueing network with a number of printers interconnected to a single service queue (the output queue). The model for POSTPROC then, is a function of the job inter-arrival times to the output queue, and the service time of the printers such that:

PRINTER RESPONSE TIME = B * C / D (eqn 13)

and

PRINTER RESPONSE TIME $>$ A ensures postprocessing stability

when

A = Distribution of inter-arrival times

B = Service rate of the printers

C = Average lines printed

D = Number of parallel printers

Recall in Chapter III it was determined that the arrival time for jobs in the system follows an exponential distribution with a mean arrival rate of .15 jobs/min. Moreover, the multiprogramming level for the system provided an effective inter-arrival time greater than the thruput time which made the system stable from job arrival to the output queue. Given this, the components of the model becomes:

A = Mean Inter-Arrival time * Multiprogramming level / min

giving

A = 6.68 * 4.28 / 60 = .47 jobs/min

and

B = Print Service Time = 1 / PRINT RATE =

1 / 1200 = .00139

C = 1,660 (see Table 6, page 44)

D = 3 (see Table 5, page 25) (eqn 14)

hence

$$\text{PRINTER RESPONSE TIME} = \frac{\text{SERVICE TIME}}{\text{MIN}} * \frac{\text{LINES}}{\text{AVG.}} / \text{PRINTERS}$$

thus

PRINTER RESPONSE TIME = .00139 * 1660 / 3 = 0.77 min/job

This contrasts to the .47 jobs arriving per minute which suggests an unstable machine. In a stable machine, departures > arrivals. A simple recalculation of the model using 5 printers provides an optimal response time for the number of jobs requiring service:

current arrival rate	=		.47 jobs/min
current response time	=	.00139 * 1660 / 3 =	.77 jobs/min
optimal response time	=	.00139 * 1660 / 5 =	.46 jobs/min

This improvement would serve to reduce the POSTPROC statistic of TURNAROUND from the current 6.53 minutes to 3.92 minutes or 40 percent, providing a 10 percent reduction in TURNAROUND time:

where

current POSTPROC = 6.53 min/job

percent of TURNAROUND = 25.0 %

and

current response time = .77 jobs/min

new response time = .46 jobs/min

then

percent POSTPROC decrease = .46 / .77 = 40 %

and

new POSTPROC = 5.53 * .40 = 3.92 min/job

while

percent decrease in TURNAROUND = 25.0 * .40 = 10 %

HYPOTHESIS 13. A Series of Internal Battlenecks are Causing the Input Queue to Overload. The test criterion established for this hypothesis could not be fully met using the INQUEUE model developed in the modeling chapter, Chapter VI. There, it was found that the variables expected to explain much of the variation in INQUEUE explained little more than 1.6 percent of the variation. An additional model, using the surrogate variables of I/O, SEVENTRK, and NINETRK for the request variables was then developed and a regression analysis made on the model. The new predictive value for INQUEUE then became:

INQUEUE = 2.28 + 6.81 (NINETRK) + 2.57 (SEVENTRK) + .35 (IO)

producing

INQUEUE = 2.28 + 6.81 (.28) + 2.57 (.39) + .35 (2.08) = 5.92

The new model provides a high correlation with the actual 6.25 observed in the sample data. Since the surrogate IO was selected as a candidate for contributing to the reduction of TURNAROUND time (see Hypothesis 1, Page 98), a case can be made that a corresponding reduction will occur in INQUEUE time if the suggested changes are made. Thus, reducing the model variables by the projected 41 percent produces:

INQUEUE = 2.28 + 6.81 (.17) + 2.57 (.23) + .35 (1.23) = 4.46

HYPOTHESIS 14. Workload will exceed capacity. To test this hypothesis, as detailed under the criterion for hypothesis testing, the current workload represented by

the sample data, was stratified by Data System to get the mean time used by each system. Furthermore, recall the initial problem statement that the smaller systems had been converted and the larger systems were yet to be added to the workload. With this constraint, only those systems which were in the top quartile of time use were selected from the population. This turned out to be 28.87 hours per month (Table 24). Based on the stated test criteria then, the machine's service requirements will exceed its capacity with the addition of 4.77 systems if none of the recommended changes are made (see Table 24).

DISCARDED HYPOTHESES

HYPOTHESIS 2. Unconverted Workload Has Caused Inefficient System Use. As stated earlier, this may be one of the reasons why the input queue is overloaded. The probable cost to test the hypothesis (see Verification Requirements Chapter V) in relation to the potential short term savings in input queue overheads time places this in the class of rejected hypotheses.

HYPOTHESIS 4. Pertains to user actions as they apply to programming practices and requires an independent investigation into how the users programs have been contructed. Again, time constraints preclude a thorough examination of the hypothesis and places it in the category of hypotheses for further study.

HYPOTHESIS 8. Refers to the placement of the operating system and can only be tested empirically, therefore this must also be placed in the class of deferred hypotheses.

HYPOTHESIS 9. Too Few Activities in the System. The verification of this hypothesis follows closely along the lines of reasoning used to verify hypothesis 5, Page 99, and is discarded in favor of that solution.

TABLE 24

CALCULATION FOR WORKLOAD CAPACITY

Maximum Additional Systems = Remaining Hours / Mean System Hours

when

Mean System Hours = 28.87

Remaining Hours = Total Capacity - Current Workload

where

Total Capacity = ((Nd * Nh) - (PM + CL)) * MP

when

Nd = days measured = 24

Nh = hours per day = 24

PM = Maintenance = (6 * 4 wks) = 24 Hrs

CL = classified = (8 * 4 wks) = 32 Hrs

MP = multiprogramming = 4.28

and

Total Capacity = ((24 * 24) - 56) * 4.28 = 2225 Hrs

also

Current Workload = mean TURNAROUND * Jobs

when

mean TURNAROUND = 24.74

Jobs = 5064

thus

Current Workload = 24.74 * 5064 = 2088 Hrs

therefore

Remaining Hours = 2225 - 2088 = 138

giving

Maximum Additional Systems = 138 / 28.87 = 4.77 (eqn 14)

HYPOTHESIS 11. Insufficient Memory. As mentioned earlier, this hypothesis requires the use of a hardware monitor to provide confidence as to its validity. Viewing the amount of CORE used by the sample jobs (Table 15B, Page 63) as a guide however, memory appears to be sufficient. This hypothesis then, is rejected.

SUMMARY

Of the initial 14 hypotheses developed to theorize the turnaround problem nine were found to be valid and testable with the tools available and within the scope of time originally defined for this thesis. Two of the discarded hypotheses were still felt to be valid assumptions pertaining to the problem and could eventually lead to a further reduction in overall turnaround time, but could not be pursued without special hardware monitors to measure resource activity during specific time intervals. These have been suggested as future research areas for possible follow-on theses efforts.

The nine retained hypotheses can all have a direct influence on the amount of time it takes a job to process through the system. A total of thirty-one percent savings in turnaround time can be achieved with the implementation of the hypothesized changes. This amounts to a net reduction of 7.67 minutes for the average job. Indeed, the reduction calculates to 647.35 raw processing hours or 6.3 full days at the current multiprogramming level. Thus, if all of the variables deemed to be a part of the TURNAROUND model were adjusted to account for the hypothesized savings, the new expected turnaround time would be:

$$\text{TURNAROUND} = .14 + 1.24\,(1.23) + 1.0\,(3.92) + 1.21\,(2.45) + .56\,(0) + 1.01\,(3.75) + .99\,(4.46) = 16.75 \text{ min/job}$$

VIII CONCLUSIONS AND RECOMMENDATIONS

CONCLUSIONS

The objectives of this thesis were to characterize the workload of the AFLC CYBER-73 computer system; to model that workload in terms of its resource usage and of its thruput in relation to the system's multiprogramming characteristics; to determine where thruput bottlenecks existed, and to develop solutions to alleviate the bottleneck conditions. The approach used was a systematic reduction of the computer's job accounting data and a rigorous statistical analysis of performance.

The DAYFILE accounting data proved to be a valuable source from which to derive descriptive statistics about where job elongation in the system could be detected. All jobs processing during the month of July, 1981, with the exception of July 13-19, were used as the data base to obtain the statistics for the evaluation. A series of reduction programs were developed to extract and parameterize variables pertaining to activities on which thruput depended. The resulting parameters were then stratified in clusters compatible with tailored statistical procedures designed to simplify data analysis of trend phenomena. Analytic models of hypothesized bottleneck areas were then developed, tuned, and tested for their accuracy in explaining the observed performance. The models were then modified to reflect hypothesized solution possibilities for the purpose of predicting future performance with theorized changes. The final results provided a clear picture as to where bottlenecks were occurring in the system and answers to what steps should be taken to solve the problem.

The analysis led to a potential 31 percent reduction in turnaround time from 24.74 minutes to 17.07 minutes per average job. Most of the theorized reduction is to be accomplished by changes in programming practices, resident file placement, operating system priority scheduling and job entry. The remaining reduction would be achieved by

the addition of two printers to relieve the bottleneck in the output queue. An additional conclusion made possible by the analysis of existing system activity and future workload enhancements led to a prediction that the machine would be saturated with the addition of 4.77 new systems if none of the changes are made.

A by-product of the evaluation and techniques used is a completely developed and tested sequence of programs and procedures which, if included in a computer program callable subroutine, could enable development program managers to quickly assess and optimize the performance of their systems before including them in the computer's workload.

This evaluation is by no means conclusive. Several hypotheses which could lead to further savings were deferred to future efforts because of unavailable time and tools. Although expected machine time savings which could result from a complete analysis of the deferred hypotheses tend to be on a diminishing returns curve as a result of partial optimization through the implementation of the previously suggested changes, full optimization should always be a goal for computer operations, and the hypotheses should not be disregarded.

RECOMMENDATIONS

AFLC should effect a series of steps to realize the 31 percent reduction in turnaround time for systems on the CYBER-73:

(1) Jobs involved in Tape and Disk I/O operations should be reviewed to ensure that the maximum blocking possible for the machine is being used. A 41 percent decrease in I/O overhead was projected through the developed models with the change.

Moreover, the change can be made with a minimum of program changes and recompiles. The result, since it only affects the number of accesses made by a program, would have no effect on the programs themselves.

(2) A command resolution should be accomplished between the user directorates to realize a more even input flow of jobs throughout the day to relieve the input queue overloading during the 0800-1200 period each day. It could be found that some systems are scheduled for running during that period in the hope that the output will be returned by the end of the primary workshift (0800-1600) when in fact, scheduling some of the jobs earlier, say during the 1600-2400 period of the day before, could accomplish the same goal.

(3) The operating system's priority scheduler should be changed to provide a longer time quantum for tape use jobs. Although heavy tape use should be discouraged in favor of disk use for optimization of I/O operations, many tape operations are unavoidable. Since this is probably the case, penalizing users of tape by excessive rollouts/swapouts is in effect degrading the overall performance of the command objective of optimum total system turnaround.

(4) Users should be prevented from using private disk packs for their systems. The private pack concept in a machine with adaquate file protection and safegards such as the CYBER only tends to add unneeded overhead to the system and the computer. It should make no difference to a user if his files are on a public or private device so long as he has reasonably unrestricted access to his data. Therefore, a switch to public devices for all user files (classified data being an exception) would permit the operating system to distribute the files for optimum accessing by user jobs.

(5) Tape files should be pre-staged. Time waiting for tapes by jobs account for 4.9 percent of turnaround time. This is the time it takes the operator to mount and assign a tape to the requesting job. If job streams (JCL) were developed such that tapes would only be used as initial input (and here only when a file could not reside on disk) and for final output while using disk for all intermediate file storage, the tape requirements could be effectively pre-assigned by the operator prior to the call by the program eliminating the wait time for jobs which must use tape for files. The small volume files which require tape residence should be included in the JCL as auxiliary operations apart from the using program and pre-staged or post-staged to permit optimum I/O when the program accesses those files. This would eliminate the need to hold a program in wait while the operator mounts and dismounts tapes.

(6) Two additional printers should be added to the computer to relieve the output queue of the backlog of jobs occurring there. Postprocessing for jobs accounts for a full 25 percent of their turnaround time. The addition of two printers would optimize the output of the current workload to the print rate of the printers instead of to the number of printers available. This change would result in an overall 220 hours per month in machine processing time.

SUMMARY

In summary, six steps are suggested to enable AFLC to realize a 31 percent reduction in job turnaround time. Forty-one percent of I/O time or 4.1 percent of turnaround time can be eliminated by maximizing I/O blocking factors. Time waiting for disk mounting can be eliminated and a corresponding 5.1 percent reduction in turnaround achieved by assigning all disk files to public devices. The printed output of jobs can be received faster along with a 10 percent reduction in turnaround time with the addition of

two printers. The elimination of tape wait time by pre/post-staging tape files provides an additional 4.9 percent reduction in turnaround time. Finally, the input queue time, accounting for 24 percent of the turnaround, will be reduced by 29 percent with the implementation of all of the recommended changes providing an additional 6.5 percent reduction in turnaround. The combined changes comes to 647 multiprogrammed computer hours per month. The time savings, if computed as a function of the multiprogramming rate and AFLC's measure of cost per computer hour, totals:*

647 / 4.28 * 12 * $220.00 = $399,084 per year

Furthermore, the savings can be computed for both Headquarters and the ALCs. This can be safely done since, as mentioned in the opening statement, the ALCs, using a similar configuration, are experiencing the same turnaround delays as Headquarters. Therefore, modifications which can improve the conditions at Headquarters can also be assumed to provide like improvements at the ALCs). Computing all sites then the yearly savings would amount to:

$399,084 * 6 (AFLC sites) = $2,394,504

(*) This cost was provided by Mr. Eldon Mongold, LMD, Oct, 198 .

BIBLIOGRAPHY

(1) AFLC. CYBER ADPE Replacement. Hq AFLC Data Automation Requirement (DAR-LOG-LMT-001). Request and justification for peripheral hardware replacement, AFLC 23 January 1981.

(2) AFLC. IBM 7080 TO COBOL (CYBER) CONVERSION STUDY. Feasibility study and documentation supporting a request for data system conversions, WPAFB, Ohio: AFLC, November 1977.

(3) Agrawala. A. K. et al. "An Approach to the Workload Characterization Problem". Computer. pp 18-32, (June 1976).

(4) Bear, J. A. Workload Characterization and Measurement of the CDC CYBER 74 Computer. Masters thesis, WPAFB, Ohio: Air Force Institute of Technology March 1977. (AD-04-184).

(5) Bell, T. E. et al. Computer Performance Analysis: Framework and initial Phases for a Performance improvement Effort. R-549-1-PR. Santa Monica, California: Rand Corporation, November 1972.

(6) Bell, T. E. Computer Performance Management Through Control Limits. TRW-55-76-01. Redondo Beach, California: TRW, January 1976.

(7) Bell, T. E. Computer Performance Analysis: Measurement Objectives and Tools. R-584-NASA/PR. Santa Monica, California: Rand Corp., February 1971.

(8) Bell, T. E. Computer Performance Analysis: Objectives and Problems in Simulating Computers. R-1051-PR. Santa Monica, California: Rand Corporation, July 1972.

(9) Benwell, Nicholas. BENCHMARKING Computer Evaluation and Measurement. Washington, D. C.: Hemisphere Publishing Corp., 1975.

(10) Blitt, J. et al. The Advanced Logistic System CYBER 73: A Simulation Model. Masters thesis, WPAFB, Ohio: Air Force Institute of Technology, December 1975. (AD-A019841).

(11) Boeing Computer Services, inc. Computer Load and Resource Analysis (*1) System Installation Manual. BCS, Seattle, Washington: BCS, July, 1976.

(12) Buzen, J. P. The Role of Computer Performance Modeling, unpublished report, Lincoln, Ma.: BGS Systems, inc., undated.

(13) CDC. SCOPE Reference Manual. Control Data CYBER 70 Computer Systems, Sunnyvale, California: CDC Corporation, 1975.

(14) CDC. Scope Integrated Scheduler Tuning Guide. Control Data CYBER 70 Computer Systems, Sunnyvale, California: CDC Corporation, 1975.

(15) CDC. CYBER 70 Model 73 System Description and Programming Information Reference Manual. Control Data CYBER 70 Series Reference Manuals, Arden Hills, Minnesota: CDC Corporation, 1971.

(16) CDC. CYBER 70 Models 72/73/74 6000 Computer Systems, Input/Output Specifications. Arden Hills, Minnesota: CDC Corporation, 1971.

(17) CDC. CYBER 70 Computer Systems Models 72, 73, 74 6000 Computer Systems, System Programmer's Reference Manual Model 72, 73, 74 Version 3.4. Arden Hills, Minnesota: CDC Corporation, 1976.

(18) Davis, Richard M. Thesis Projects in Science and Engineering. New York: St. Martins Press, 1980.

(19) Dodson, Philip O. Control Data Corporations CYBER 70: Procedures for Performance Evaluation. Masters Thesis, WPAFB, Ohio: Air Force Institute of Technology, 1974. (AD-785129).

(20) Ferrari, Domenico. Computer Systems Performance Evaluation. Englewood Cliffs, N. J.: Prentice-Hall, 1978.

(21) Goldberg, Robert et al. "bottlenecks in Operating Systems," Symposium on Computer Performance. EDP Performance Review. Phoenix, Arizona: Applied Computer Research, 1976.

(22) Hartrum, Thomas C. et al. Computer Performance Evaluation readings. Unpublished compilation of EE 7.52 articles, WPAFB, Ohio: School of Engineering, Air Force Institution of Technology, 1981.

(23) Hartrum, Thomas C. Development of A Computer Turnaround Time Model, Unpublished Report, June 1979.

(24) Helleman, H. and T. F. Conroy. Computer System Performance. New York: McGraw-Hill Book Co., 1975.

(25) Highland, Harold J. et al. Proceedings of Symposium on Simulation of Computer Systems. August 12-14 Bolder Colorado: National Bureau of Standards, 1975.

(26) Howard, Philip C. Highlights of the Computer Performance Evaluation users group conference, EDP Performance Review, Phoenix, Arizona: Applied Computer Research Vol 5, No. 10, PP 2-6 (November 1977).

(27) Joseph, Gilbert W. Guidelines for a Capacity Usage Study, Unpublished report, Washington, D. C.: Federal Simulation Center, undated.

(28) Kiviat , P. J. et al. "Software Unit Profiles and Kiviat Figures". Performance Evaluation Review. Vol 2, No. 3 PP 2-12, (September 1973).

(29) Lewis, Paul C. A Computer Performance Evaluation Educational Tool. WPAFB, Ohio: Air Force Institute of Technology, 1977.

(30) LMO. HQ AFLC CYBER-73 "Production Master Schedule", Unpublished annual computer schedule by Data System, Frequency, Time and Core use, Hq AFLC/LMO, May 1981.

(31) LMT. Request for Computer Evaluation Study. Letter to AFIT School of Engineering, HQ AFLC/LMT, December 1980.

(32) Motsch, Bob and Tom Lehman. "Picturing Computer Activity." EDP Performance Review. Phoenix, Arizona: Applied Computer Research, Vol. 8, No. 4 (April 1980).

(33) Muntz, R. R. "Analytic Modeling of Interactive Systems". Proceedings of the IEEE. PP 946-953 (June 1975).

(34) Nie, Norman H. et al. Statistical Package for the Social Sciences. New York: McGraw-Hill Book Company, Inc., 1979.

(35) Svobodova, L. Computer Performance Measurement and Evaluation Methods: Analysis and Implications. New York: American Elsevier Publishing Co., 1976.

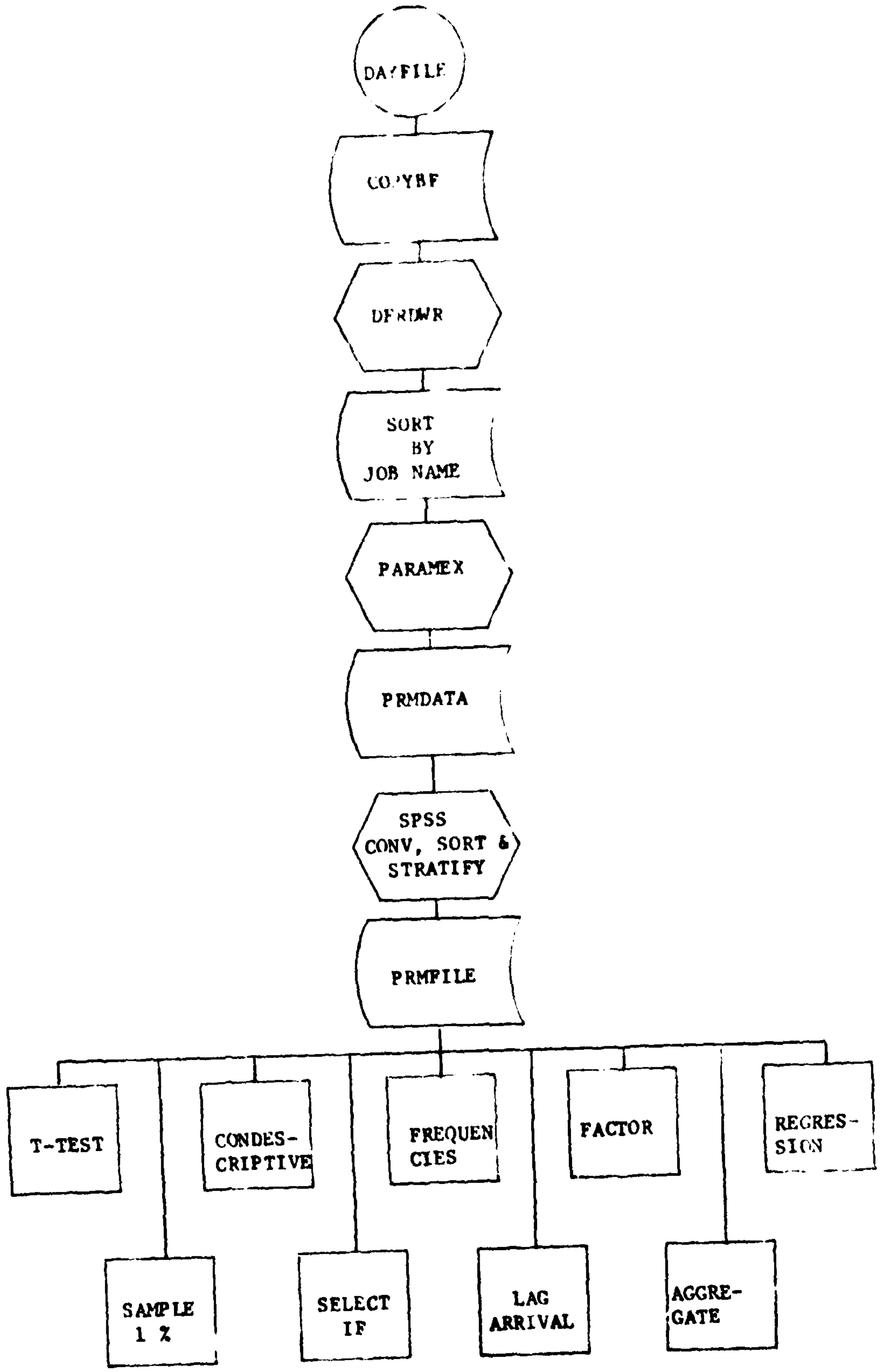
(36) Thompson, J. W. Computer Performance Evaluation of Individual Workloads on the ASD CDC CYBER System. Masters thesis, WPAFB, Ohio: Air Force Institute of Technology, 1978. (AD-A064313).

(37) Wallack, Barry M. et al. WWMCCS system Tuning Process. Washington, D. C.: Worldwide Military Command and Control System (WWMCCS), February 1978.

(38) Watson, R. A. Computer Performance Analysis: Applications of Accounting Data. R-573-NASA/PR Santa Monica, California. Rand Corporation, May 1971.

APPENDIX A

PROGRAM LISTINGS



JOB FLOW SEQUENCE FOR DAYFILE DATA FROM RAW DATA TO SPSS OUTPUT

```
PROGRAM PARAMEX(INPUT,PFILE,PRMDECK,OUTPUT);

(*****************************************************)
(*****************************************************)
   (*PARAMEX (PARAMETER EXTRACT) BUILDS A STATISTICAL
   RECORD OF THE ACTIVITIES OF A JOB AS IT PROCESSES
   THROUGH A COMPUTER SYSTEM. THE CYBER USES A DAYFILE
   PROCEDURE TO TRACE A JOB FROM INPUT (CARD READ)
   THROUGH OUTPUT (PRINTING OR PUNCHING), NOTING EACH
   JOB STEP ACTIVITY AS IT OCCURS. STATISTICS OF THE
   ACTIVITIES ARE CAPTURED ON THE DAYFILE TAPE AND
   INCLUDES THE TIME OF THE OCCURANCE. THIS RUN USES
   A SORTED, REDUCED (FROM A PREVIOUS PASCAL PROGRAM)
   VERSION OF THE DAYFILE TO EXTRACT THE PARAMETERS AND
   BUILDS 2, 102 CHARACTER RECORDS FOR EACH JOB RUN.
   INCLUDED ON THE OUTPUT RECORDS ARE:

        PARAMETER      DESCRIPTION               RECORD     COLUMNS

        NAME           JOB NAME                    1          1- 8
        DAY            DAY RUN                     1         10-11
        ARRIVAL        TIME OF JOB ARRIVAL         1         13-19
        DISKWAIT       DISK WAIT TIME              1         21-27
        THRUPUT        PROCESSING TIME             1         29-35
        SORTXTIME      TIME SORTING                1         37-43
        REQ7           TAPE REQ - 7 TRK            1         45-46
        REQ9           TAPE REQ - 9 TRK            1         48-49
        TREQ           TIME REQUESTED              1         51-54
        CMREQ          CM REQUESTED                1         56-61
        DISKIO         DISK ACCESSES               1         63-68
        MEM            CORE USED                   1         70-75
        MAXMEM         MAX MEM USED                1         77-82
        TAPEWAIT       TAPE WAITING TIME           1         84-90
        PPUSAGE        PP USAGE TIME               2          2-11
        SS             SYSTEM SECONDS              2         12-21
        CPU            CPU TIME USED               2         23-30
        CM             CM USED                     2         32-39
        IO             I/O USED                    2         41-48
        LINES          LINES PRINTED               2         50-56
        RCDSORTED      RCDS SORTED                 2         58-63
        TAPEIO         TAPE BLOCKS READ/WRITTEN    2         65-71
        SEVENTR        7 TRK TAPE USE              2         73-74
        NINETR         9 TRK TAPE USE              2         76-77
        POSTPROC       POST PROCESSING             2         79-85
        INQUEUE        INPUT QUEUE TIME            2         87-93
        TURNAROUND     TURNAROUND TIME             2         95-101

                                                                  *)
(*****************************************************)
```

```
CONST
BLANKS = '              ';
MAXC = 80;
MAXD = 200;
SPACE = ' ';
TYPE
PARAMDATA = PACKED ARRAY [1..30] OF CHAR;
DATALINE = PACKED ARRAY[1..MAXD] OF CHAR;
SIXCHAR = PACKED ARRAY[1..6] OF CHAR;
FOURCHAR = PACKED ARRAY[1..4] OF CHAR;
PARAMLOC = PACKED ARRAY[1..12] OF CHAR;
TWOCHAR = PACKED ARRAY[1..2] OF CHAR;
PARAMFIELDS =
RECORD
TSC : PARAMLOC;
NTAC : PARAMLOC;
MTAC : PARAMLOC;
WRC : PARAMLOC;
LPC : PARAMLOC;
CRC . PARAMLOC;
EQC : PARAMLOC;
MTC : PARAMLOC;
MDC : PARAMLOC;
ALC : PARAMLOC;
STC : PARAMLOC;
LBC : PARAMLOC;
RQC : PARAMLOC;
END;
TIMEO = REAL;
VAR
TIMEREQ : PARAMLOC;
NAME : PARAMLOC;
DAY : TWOCHAR;
ARRIVAL : TIMEO;
CPU : PARAMLOC;
MEM : PARAMLOC;
MEMMAX : PARAMLOC;
CM : PARAMLOC;
IO : PARAMLOC;
PP : PARAMLOC;
SWAP : PARAMLOC;
DISKWAIT : TIMEO;
DWSTART : TIMEO;
DISKIO : PARAMLOC;
TAPEIO : INTEGER;
INQUEUE : TIMEO;
SEVENTR : INTEGER;
NINETR : INTEGER;
SEVENREQ : TWOCHAR;
NINEREQ : TWOCHAR;
CMREQ : PARAMLOC;
```

```
STARTS : TIMEO;
POSPROC : TIMEO;
TAPEWAIT : TIMEO;
STPOST : TIMEO;
TWSTART : TIMEO;
   TURNAROUND : TIMEO;
SORTATIME : TIMEO;
RCDSSORTED : PARAMLOC;
LINES : PARAMLOC;
THRUPUT : TIMEO;
SWITCH : BOOLEAN;

(**************************************************)

   (*TRUE WHEN EOJ CARD HAS BEEN ENCOUNTERED
     FOR THE CURRENT JOB                        *)

(**************************************************)



KEY : INTEGER;

(**************************************************)

   (*USED TO NOTE THE NUMBER OF SUCCESSFUL
     CHARACTER COMPARES WHEN COMPARING
     AN INPUT RECORD AGAINST THE VARIABLE

(**************************************************)

     KEYS - BLOCKS, MOUNT, ETC.          *)

J : INTEGER;
PARAMFILE : PARAMFIELDS;
DLINE : DATALINE;
   (*ONE FULL LINE OF DATA*)

PRMDECK : TEXT;
PFILE : TEXT;
BUFFER : PARAMDATA;
I : INTEGER;
CH : CHAR;
DFDATA : DATALINE;


PROCEDURE WRITEPARAMS;
```

```
(*******************************************************)

   (*BUILD PARAMETER-KEY FILE FOR MATCHING AGAINST
     INPUT RECORDS (DAYFILL RECORDS ARE UNIQUE FOR
     THESE LOCATIONS). THUS, DUE TO THE INABILITY
     IN PASCAL TO INITIALIZE ARRAYS WITH CONSTANTS,
     STRING CONSTANTS MUST FIRST BE WRITTEN TO
     DISK THEN RE-READ AS CHARACTERS TO BUILD THE
     ARRAYS.                                         *)

(*******************************************************)

BEGIN
WRITELN(PR4DECK,'   ** TOTAL R',BLANKS);
WRITELN(PR4DECK,' ( J          ',BLANKS);
WRITELN(PR4DECK,' ( M          ',BLANKS);
WRITELN(PR4DECK,' BLOCKS       ',BLANKS);
WRITELN(PR4DECK,'INES PR       ',BLANKS);
WRITELN(PR4DECK,'FRO4          ',BLANKS);
WRITELN(PR4DECK,'  ENTERED IN',BLANKS);
WRITELN(PRMDECK,' MOUNT,       ',BLANKS);
WRITELN(PR4DECK,' MOUNTE       ',BLANKS);
WRITELN(PR4DECK,' ALREAD       ',BLANKS);
WRITELN(PR4DECK,' SORTMR       ',BLANKS);
WRITELN(PR4DECK,' LABEL        ',BLANKS);
WRITELN(PR4DECK,' REQUEST      ',BLANKS);
END;

PROCEDURE READPARAMS;

(*******************************************************)

   (*READ THE PARAMETER-KEY FILE, STORE IN THE 12
     CHARACTER BUFFER PROVIDED BY INITIALIZE.
     EXECUTED ONCE FOR EACH OF THE 13 KEYS       *)

(*******************************************************)

VAR J : INTEGER;
BEGIN
J := 0;
REPEAT
READ(PR4DECK,CH);
J := J + 1; BUFFER[J] := CH
UNTIL EOLN(PR4DECK);
READLN(PR4DECK)
END;
PROCEDURE INITIALIZE(VAR  PARAM:PARAMLOC);
```

```
(**********************************************

   (*[illegible] THE KEYS FROM THE PREVIOUSLY WRITTEN
     [illegible] FILE. PARAM IS ONE OF 13 LOCATIONS TO
     HOLD THE KEYS. CALLED FROM GETPARAMS          *)

(***********************************************************)

BEGIN
READPARAMS,
FOR I := 1 TO 12 DO
BEGIN
PARAM[I] .= BUFFER[I]
END;
END;
PROCEDURE GETPARAMS;

   (*CALLED FROM MAIN TO RETRIEVE THE PARAMETER KEYS
     BUILT AND STORED ON DISK AT THE BEGINNING OF JOB.
     KEY DESCRIPTIONS ARE:                             *)

BEGIN
WITH PARAMFILE DO BEGIN
INITIALIZE(TSC);    (*TOTAL RECORDS SORTED*)
INITIALIZE(NTAC);   (*NINETRACK TAPE ASSIGNMENT*)
INITIALIZE(MTAC);   (*SEVEN TRACK TAPE ASSIGNMENT*)
INITIALIZE(WRC);    (*TAPE BLOCKS READ/WRITTEN*)
INITIALIZE(LPC);    (*LINES PRINTED*)
INITIALIZE(CRC);    (*CARDS READ*)
INITIALIZE(EQC);    (*JOB ENTERED INPUT QUEUE*)
INITIALIZE(MTC);    (*DISK MOUNT DIRECTIVE*)
INITIALIZE(MDC);    (*MOUNT ACKNOWLEDGEMENT*)
INITIALIZE(ALC);    (*ALREADY MOUNTED MSSG*)
INITIALIZE(STC);    (*SORT DIRECTIVE*)
INITIALIZE(LBC);    (*TAPE REQUEST(LABEL)*)
INITIALIZE(RQC);    (*TAPE REQUEST(REQUEST)*)
END;
END;
FUNCTION TIME:REAL;

   (*CONVERT TIME FROM CHARACTER VALUE TO REAL*)

VAR T : REAL;
BEGIN
T := 0;
T := T + (ORD(DLINE[2])-ORD('0'))*10;
T := T + (ORD(DLINE[3])-ORD('0'))*1;
T := T + (ORD(DLINE[5])-ORD('0'))*(0.1);
T := T + (ORD(DLINE[6])-ORD('0'))*(0.01);
```

```
T := T + (ORD(DLINE[8])-ORD('0'))*(0.001);
T := T + (ORD(DLINE[9])-ORD('0'))*(0.0001)
TIME := T;
END;


PROCEDURE GETNAMEANDARRIVAL;

   (*GET JOB NAME, ARRIVAL TIME, AND START OF QUEUE TIME*)

   BEGIN
J := 11;
FOR I := 1 TO 8 DO BEGIN
NAME[I] := DLINE[J];
J := J + 1;
END;
ARRIVAL := TIME;
INQUEUE := ARRIVAL - INQUEUE;
END;
FUNCTION COMPLOC7 (VAR CLOC : PARAMLOC): BOOLEAN;

   (*SEE IF CURRENT INPUT RECORD IS AN ARRIVING
     JOB OR LINES PRINTED STATISTIC, RETURN TRUE
     IF YES.                                   *)

BEGIN
J:=0; KEY:=0;
FOR I := 30 TO 36 DO BEGIN
J := J + 1;
IF DLINE[I] = CLOC[J] THEN
KEY := KEY + 1
END;
IF KEY = 7 THEN
COMPLOC7 := TRUE
ELSE COMPLOC7 := FALSE
END;
PROCEDURE GETDOLLARSTATS;

   (*THIS PROCEDURE GROUP TAKES THE SYSTEM GENERATED
     STATISTICS (NOTED BY $ IN CC-20) AND BUILDS THE
     JOB CARD AND RESOURCE USE PARAMETERS.         *)

PROCEDURE TAPEREQ;

   (*GET SEVEN AND NINE TRACK REQUEST STATISTICS   *)

BEGIN
SEVENREQ[1] := DLINE[31];
SEVENREQ[2] := DLINE[32];
NINEREQ[1] := DLINE[41];
```

```
[illegible] := DLINE[42];
END;
PROCEDURE TIMEREQST;

    (*GET TIME REQUESTED FROM JOB CARD*)



BEGIN
TIMEREQ[1] := DLINE[37];
TIMEREQ[2] := DLINE[38];
TIMEREQ[3] := DLINE[39];
TIMEREQ[4] := DLINE[40];
J := 57;
FOR I := 1 TO 6 DO BEGIN
CAREQ[I] := DLINE[J];
J:= J+1
END
END;
PROCEDURE GETACCESS;

    (*GET NUMBER OF DISK ACCESSES MADE*)

BEGIN
J := 32;
FOR I := 1 TO 6 DO BEGIN
J := J+1;
DISKIO[I] := DLINE[J];
END;
END;
PROCEDURE COREUSE;

    (*GET CORE AND MEMORY RESOURCES USED*)

VAR K:INTEGER;
BEGIN
J := 24; K := 40;
FOR I := 1 TO 6 DO BEGIN
J:=J+1; K:=K+1;
MEM[I] := DLINE[J];
MEMMAX[I] := DLINE[K];
END;
END;
PROCEDURE PPUSE;

    (*GET PERIPHERAL PROCESSOR USE TIME*)

BEGIN
J:=24;
```

```
FOR I:= 1 TO 10 DO BEGIN
J:=J+1;
M[I]:= DLINE[J];
END;
DAY[1] := DLINE[53];
DAY[2] := DLINE[54];
END;
PROCEDURE SWAPOUT;

    (*GET TOTAL SYSTEM SECONDS USED*)

    (*SYSTEM SECONDS CONSISTS OF WEIGHTED
      CPU, CM & I/O TIME USED           *)

BEGIN
IF DLINE[22] = 'S' THEN BEGIN
J:=42;
FOR I:= 1 TO 9 DO BEGIN
J:=J+1;
SWAP[I]:= DLINE[J];
END;
END;
END;
PROCEDURE CPUUSE;

    (*GET CENTRAL PROCESSOR'USAGE TIME*)

BEGIN
J:=25;
FOR I:= 1 TO 8 DO BEGIN
J:= J+1;
CPU[I] := DLINE[J];
END;
END;
PROCEDURE CMUSE;

    (*GET CENTRAL MEMORY WORDS USED*)

BEGIN
J:= 42;
FOR I := 1 TO 8 DO BEGIN
J := J+1;
CM[I] := DLINE[J];
END;
END;
PROCEDURE IOUSE;

    (*GET I/O TIME USED*)

BEGIN
```

```
J := 42;
FOR I := 1 TO 6 DO BEGIN
J := J +1;
IO[I] := DLINE[J];
END;
END;
BEGIN (*GETDOLLARSTATS*)

    (*TEST CC 21-23 FOR KEY TO WHICH SUBROUTINE
      TO ENTER. THESE COLUMNS ARE UNIQUE FOR
      EACH RECORD AFTER A $ HAS BEEN ENCOUNTERED
      IN CC-20. THE $-TYPE RECORDS ARE BUILT
      BY THE DAYFILE FOR EACH JOB ENTERING OR
      EXITING THE SYSTEM.                      *)

IF COMPLOC7(PARAMFILE.CRC) = TRUE THEN GETNAMEANDARRIVAL
ELSE
IF DLINE[21] = 'A' THEN
BEGIN
IF DLINE[23] = 'C' THEN GETACCESS
ELSE BEGIN
IF DLINE[25] = 'M' THEN TAPEREQ
ELSE TIMEREQST
END
END
ELSE
IF DLINE[21] = 'M' THEN COREUSE
ELSE
IF DLINE[21] = 'P' THEN PPUSE
ELSE
IF DLINE[21] = 'S' THEN SWAPOUT
ELSE
IF DLINE[21] = 'E' THEN
BEGIN
IF ARRIVAL <> 0 THEN
BEGIN
TURNAROUND := TIME - ARRIVAL;
STPOST := TIME;
SWITCH := TRUE
END
END
ELSE
IF DLINE[21] = 'C' THEN
BEGIN
IF DLINE[22] = 'P' THEN CPUUSE
ELSE CMUSE
END
ELSE
IF DLINE[21] = 'I' THEN
BEGIN
```

```
IF DLINE[22] = 'O' THEN IOUSE
END
END; (*GETDOLLARSTATS*)
PROCEDURE TOTSORT;

    (*GET TOTAL RECORDS SORTED AND THE
      TIME THE SORT COMPLETED.        *)

BEGIN
SORTXTIME := SORTXTIME + (TIME-STARTS);
J := 52,
FOR I := 1 TO 6 DO BEGIN
J := J+1;
IF DLINE[J] <> '*' THEN
RCDSSORTED[I] := DLINE[J]
END;
END;
PROCEDURE BLOCKS;

    (*GET NUMBER OF TAPE BLOCKS READ/WRITTEN.
      CONVERT FROM CHARACTER TO INTEGER VALUE.
      THEN ADD THE VALUE TO THE CUMULATIVE
      TAPEIO VALUE UNTIL A NEW JOB ARRIVES *)

VAR TP : INTEGER;
BEGIN
TP := 0;
TP := TP+(ORD(DLINE[43])-ORD('0'))*100000;
TP := TP+ (ORD(DLINE[44])-ORD('0'))*10000;
TP := TP+ (ORD(DLINE[45])-ORD('0'))*1000;
TP := TP+ (ORD(DLINE[46])-ORD('0'))*100;
TP := TP+ (ORD(DLINE[47])-ORD('0'))*10;
TP := TP+ (ORD(DLINE[48])-ORD('0'))*1;
TAPEIO := TAPEIO + TP;
END;
PROCEDURE LINESP;

    (*GET THE NUMBER OF LINES PRINTED AND NOTE
      THE TIME OF JOB COMPLETION (THIS IS THE
      ENDING TIME USED TO COMPUTE TURNAROUND.*)

BEGIN
J := 21;
FOR I := 1 TO 7 DO BEGIN
LINES[I] := DLINE[J];
J := J + 1;
END;
THRUPUT := TIME - INQUEUE
END;
```

```
PROCEDURE TAPEWAITTIME;

    (*GET THE TIME WHEN A REQUESTED TAPE WAS
      MADE READY. ADD TO TOTAL TAPE WAIT
      TIME FOR THIS JOB. ZERO OUT TAPEWAIT
      WHEN A NEW JOB ARRIVES                *)

VAR T : TIMEO;
BEGIN
T := TIME;
IF TWSTART <> 0 THEN
TAPEWAIT := TAPEWAIT + (T - TWSTART);
TWSTART := 0;
END;
PROCEDURE DISKWAITTIME;

    (*ZERO OUT ALL VARIABLE LOCATIONS. EXECUTED
      AFTER WRITING PARAMETERS FOR A PREVIOUS
      JOB OR UPON ARRIVAL OF A NEW JOB BEFORE
      EOJ IS ENCOUNTERED FOR THE PREVIOUS JOB. *)

VAR T . TIMEO;
BEGIN
T := TIME;
IF DWSTART <> 0 THEN
DISKWAIT := DISKWAIT + (T-DWSTART);
END;
PROCEDURE CLEARPARAM;
BEGIN
FOR I .= 1 TO 12 DO
BEGIN
NAME[I] := SPACE;
RCDSSORTED[I] := '0';
TIMEREQ[I] := '0';
CPU[I] := '0';
MEMMAX[I] := '0';
SWAP[I] := '0';
DISKIO[I] := '0';
LINES[I] := '0';
CMREQ[I] := '0';
TIMEREQ[I] := '0';
LINES[I] := '0';
MEM[I] := '0';
CM[I] := '0';
IO[I] := '0';
PP[I] := '0';
END;
SORTXTIME := 0;
TURNAROUND := 0;
ARRIVAL := 0;
```

```
TAPEWAIT := 0;
POSPROC := 0;
STPOST := 0;
TWSTART := 0;
THRUPUT := 0;
DISKWAIT := 0;
INQUEUE := 0;
TAPEIO := 0;
DWSTART := 0;
STARTS := 0;
NINETR := 0
SEVENTR := 0;
FOR I := 1 TO 2 DO BEGIN
SEVENREQ[I] := '0';
DAY[I] := '0';
NINEREQ[I] := '0';
END;
END;
FUNCTION NAMEIN:BOOLEAN;

    (*COMPARE NAME OF JOB ASSOCIATED WITH THE
      ARRIVING RECORD WITH THE NAME OF THE LAST
      RECORD. RETURN FALSE IF NAMES ARE DIFFERENT.
      (FALSE INDICATES A PARTIAL JOB WAS PREVIOUSLY
      BEING BUILT. PARTIAL JOBS ARE IGNORED)     *)

BEGIN
J := 0; KEY := 0;
FOR I := 11 TO 18 DO BEGIN
J := J + 1;
IF DLINE[I] = NAME[J] THEN
KEY := KEY + 1
END;
IF KEY = 8 THEN
NAMEIN := TRUE
ELSE NAMEIN := FALSE
END;


PROCEDURE SAVEDATA;

    (*EXECUTED WHEN A NEW JOB ARRIVAL OCCURS OR
      AFTER A LINES-PRINTED RECORD IS DETECTED
      AND USED. OUTPUT 2, 102 CHARACTER RECORDS
      PER PARAMETERIZED JOB. CLEAR THE VARIABLE
      FIELDS WHEN COMPLETE.                     *)

BEGIN
SWITCH := FALSE;
IF NAME[2] <> ' ' THEN
```

```
IF TURNAROUND <> 0 THEN BEGIN
FOR I := 1 TO 3 DO
WRITE(PFILE,NAME[I]);
WRITE(PFILE,SPACE,DAY[1],DAY[2],SPACE,ARRIVAL:7:4,SPACE);
WRITE(PFILE,DISKWAIT:7:4,SPACE,TURNAROUND:7:4,SPACE,SORTXTIME:7:
4,SPACE);
WRITE(PFILE,SEVENREQ[1],SEVENREQ[2],SPACE);
WRITE(PFILE,NINEREQ[1],NINEREQ[2],SPACE);
FOR I := 1 TO 4 DO
WRITE(PFILE,TIMEREQ[I]);
WRITE(PFILE,SPACE);
FOR I := 1 TO 6 DO
WRITE(PFILE,CMREQ[I]);
WRITE(PFILE,SPACE);
FOR I := 1 TO 6 DO
WRITE(PFILE,DISKIO[I]);
WRITE(PFILE,SPACE);
FOR I := 1 TO 6 DO
WRITE(PFILE,MEM[I]);
WRITE(PFILE,SPACE);
FOR I := 1 TO 6 DO
WRITE(PFILE, MEMMAX[I]);
WRITELN(PFILE,SPACE,TAPEWAIT:7:4,SPACE);
WRITE(PFILE,SPACE);
FOR I := 1 TO 10 DO
WRITE(PFILE,PP[I]);
WRITE(PFILE,SPACE);
FOR I := 1 TO 9 DO
WRITE(PFILE,SWAP[I]);
WRITE(PFILE,SPACE);
FOR I := 1 TO 8 DO
WRITE(PFILE,CPU[I]);
WRITE(PFILE,SPACE);
FOR I := 1 TO 8 DO
WRITE(PFILE,CM[I]);
WRITE(PFILE,SPACE);
FOR I := 1 TO 8 DO
WRITE(PFILE,IO[I]);
WRITE(PFILE,SPACE);
FOR I := 1 TO 7 DO
WRITE(PFILE,LINES[I]);
WRITE(PFILE,SPACE);
FOR I := 1 TO 6 DO
WRITE(PFILE,RCDSSORTED[I]);
WRITE(PFILE,SPACE,TAPEIO:7,SEVENTR:3,NINETR:3,SPACE);
WRITE(PFILE,POSPROC:7:4,SPACE,INQUEUE:7:4,SPACE);
WRITELN(PFILE,THRUPUT:7:4);
END;
CLEARPARAM;
END;
FUNCTION COMPLOC4 (VAR CLOC : PARAMLOC):BOOLEAN;
```

```
    (*SEE IF CURRENT RECORD DENOTES SEVEN OR NINE
      TRACK TAPE ASSIGNMENT                          *)

BEGIN
J:=0; KEY:=0;
FOR I := 20 TO 23 DO BEGIN
J := J + 1;
IF DLINE[I] = CLOC[J] THEN
KEY := KEY + 1
END;
IF KEY = 4 THEN
COMPLOC4 := TRUE
ELSE COMPLOC4 := FALSE
END;
FUNCTION COMPLOC5 (VAR CLOC:PARAMLOC):BOOLEAN;

    (*RETURN TRUE IF CURRENT RECORD DENOTES TAPE
      BLOCKS WERE READ OR WRITTEN                 *)

BEGIN
J:=0; KEY:=0;
FOR I := 26 TO 33 DO BEGIN
J := J + 1;
IF DLINE[I] = CLOC[J] THEN
KEY := KEY + 1
END;
IF KEY = 8 THEN
COMPLOC5 := TRUE
ELSE COMPLOC5 := FALSE
END;
FUNCTION COMPLOC6 (VAR CLOC:PARAMLOC):BOOLEAN;

    (*TESTING FOR (1) DISK MOUNTING
                  (2) DISK MOUNT ACKNOWLEDGEMENT
                  (3) REQUEST FOR TAPE DRIVE
                  (4) START OF A SORT          *)

BEGIN
J:=0; KEY:=0;
FOR I := 20 TO 26 DO BEGIN
J := J + 1;
IF DLINE[I] = CLOC[J] THEN
KEY := KEY + 1
END;
IF KEY = 7 THEN
COMPLOC6 := TRUE
ELSE COMPLOC6 := FALSE
END;
FUNCTION COMPLOC12 (VAR CLOC:PARAMLOC):BOOLEAN;
```

```
    (*RETURN TRUE IF CURRENT RECORD NOTES A JOB
      ARRIVAL TO THE INPUT QUEUE                *)

BEGIN
J:=0; KEY:=0;
FOR I := 20 TO 31 DO BEGIN
J := J + 1;
IF DLINE[I] = CLOC[J] THEN
KEY := KEY + 1
END;
IF KEY = 12 THEN
COMPLOC12 := TRUE
ELSE COMPLOC12 := FALSE
END;


PROCEDURE GETDFDATA;

    (*READ INPUT RECORD. STORE IN COMMON HOLD, DLINE*)

BEGIN
I := 0;
FOR J := 1 TO MAXD DO
DLINE[J] := SPACE;
WHILE NOT EOLN DO
BEGIN
I := I + 1;
READ(CH);
DLINE[I] := CH
END;
READLN
END;
PROCEDURE GETLINE;

    (*DETERMINE WHICH KIND OF RECORD WAS READ. CHECK
      NAME OF JOB. IF NEW, WRITE PARAMETERIZED RECORD
      JUST BUILT. IF SAME, DETERMINE ITS TYPE AND
      GO TO THE APPROPRIATE SUBROUTINE.             *)

BEGIN
I := 0;
J := 0;
IF NAMEIN = FALSE THEN SAVEDATA;
IF SWITCH = TRUE THEN BEGIN
IF COMPLOC7(PARAMFILE.LPC) = TRUE THEN
LINESP
ELSE THRUPUT := TURNAROUND;
POSPROC := TIME - STPOST;
SAVEDATA
```

```
END;
IF DLINE[20] = '$' THEN GETDOLLARSTATS
ELSE BEGIN
WITH PARAMFILE DO
BEGIN
IF COMPLOC4(NTAC) = TRUE THEN
BEGIN
NINETR := NINETR + 1;
TAPEWAITTIME
END
ELSE IF COMPLOC4(MTAC) = TRUE THEN
BEGIN
SEVENTR := SEVENTR + 1;
TAPEWAITTIME
END
ELSE
BEGIN
IF COMPLOC6(MTC) = TRUE THEN DWSTART:=TIME
ELSE IF COMPLOC6(MDC) = TRUE THEN DISKWAITTIME
ELSE IF COMPLOC6(ALC) = TRUE THEN DISKWAITTIME
ELSE IF COMPLOC6(STC) = TRUE THEN STARTS:=TIME
ELSE BEGIN
IF COMPLOC12(TSC) = TRUE THEN TOTSORT
ELSE IF COMPLOC12(EQC) = TRUE THEN INQUEUE := TIME
ELSE IF COMPLOC5(WRC) = TRUE THEN BLOCKS
ELSE BEGIN
IF COMPLOC6(LBC) = TRUE THEN TWSTART := TIME
ELSE IF COMPLOC6(RQC) = TRUE THEN TWSTART := TIME
END
END
END
END
END
END;


BEGIN (*MAIN*)
REWRITE(PRMDECK);
WRITEPARAMS;
REWRITE(PFILE);
RESET(PRMDECK);
GETPARAMS;
CLEARPARAM;
WHILE NOT EOF DO
BEGIN
GETDFDATA;
GETLINE
END
END. (*MAIN*)
```

```
PROGRAM DFREDWR(INPUT,REDUCED,PRMDECK,OUTPUT);

(*********************************************************************)

   (*DFREDWR (DAYFILE REDUCTION WEEKLY RUN) READS A WEEKLY DAY-
   FILE DATA TAPE TO EXTRACT MESSAGES PERTAINING TO JOB FLOW
   ACTIVITY IN THE COMPUTER SYSTEM. BASICALLY, THE PROGRAM EX-
   TRACTS ALL RECORDS CONTAINING A "$" IN COLUMN 20 OF THE INPUT
   RECORD (THESE ARE SYSTEM GENERATED MESSAGES SIGNALING INCOM-
   MING AND OUTGOING JOBS, THEIR RESOURCE REQUESTS AND USAGE, AND
   THE TIME OF DAY THE JOB PROCESSED) AND PROGRAM GENERATED MES-
   SAGES PERTAINING TO RESOURCES. KEY PARAMETERS SEARCHED ARE:

       PARAMETER    DESCRIPTION                        PSEUDO NAME

       $            SYSTEM MESSAGES TO START AND END JOB NONE
       ** TOTAL R   TOTAL RECORDS SORTED               TSC
       ( N          9 TRACK TAPE ASSIGNMENT            NTAC
       ( M          7 TRACK TAPE ASSIGNMENT            MTAC
       BLOCKS       TAPE BLOCKS READ/WRITTEN           WRC
       ES PRIN      TOTAL LINES PRINTED BY JOB         LPC
       DS READ      CONTROL CARDS READ BY SYSTEM       CRC
       ENTERED IN   JOB ENTERED INPUT QUEUE            EQC
       MOUNT        REQUEST FOR DISK PACK              MTC
       ALREA        DISK MOUNT ACK BY SYSTEM           ALC
       SORTMR       START OF SORT                      STC
       LABEL        REQUEST FOR TAPE                   LBC
       REQUEST      REQ FOR TAPE OR DISK FILE          RQC

                                                                   *)
(*********************************************************************)

CONST
BLANKS = '              ';
MAXC = 80;
MAXD = 220;
SPACE = ' ';
TYPE
PARAMDATA = PACKED ARRAY [1..80] OF CHAR;
DATALINE = PACKED ARRAY[1..MAXD] OF CHAR;
PARAMLOC = PACKED ARRAY[1..12] OF CHAR;
...
```

```
PARAMFIELDS =
RECORD
TSC : PARAMLOC;
NTAC : PARAMLOC;
MTAC . PARAMLOC;
WRC : PARAMLOC;
LPC : PARAMLOC;
CRC : PARAMLOC;
EQC : PARAMLOC;
MTC : PARAMLOC;
ALC : PARAMLOC;
STC : PARAMLOC;
LBC : PARAMLOC;
RQC : PARAMLOC;
END;
VAR
KEY : INTEGER;
PFILE : TEXT;
J : INTEGER;
PARAMFILE : PARAMFIELDS;
DLINE : DATALINE;
PRMDECK : TEXT;
REDUCED : TEXT;
BUFFER : PARAMDATA;
I : INTEGER;
CH : CHAR;
DFDATA : DATALINE;


PROCEDURE WRITEPARAMS;

(*****************************************************************
   DUE TO THE INABILITY IN PASCAL TO INITIALIZE ARRAYS WITH
   CONSTANTS, STRING CONSTANTS MUST FIRST BE WRITTEN TO DISK
   THEN RE-READ AS CHARACTERS TO BUILD THE ARRAYS. THIS ROU-
   TINE, CALLED AT THE START OF THE PROGRAM, WRITES THE STRING
   ARRAYS TO DISK.
*****************************************************************)

BEGIN
WRITELN(PRMDECK,'   ** TOTAL R',BLANKS);
WRITELN(PRMDECK,' ( N         ',BLANKS);
WRITELN(PRMDECK,' ( M         ',BLANKS);
WRITELN(PRMDECK,' BLOCKS      ',BLANKS);
WRITELN(PRMDECK,'ES PRIN      ',BLANKS);
WRITELN(PRMDECK,'DS READ      ',BLANKS);
WRITELN(PRMDECK,'  ENTERED IN',BLANKS);
WRITELN(PRMDECK,' MOUNT       ',BLANKS);
WRITELN(PRMDECK,' ALREA       ',BLANKS);
WRITELN(PRMDECK,' SORTMR      ',BLANKS);
..
```

```
WRITELN(PRMDECK,' LABEL        ',BLANKS);
WRITELN(PRMDECK,' REQUEST      ',BLANKS);
END;

PROCEDURE READPARAMS;

(*****************************************************************
   THIS PROCEDURE READS THE PARAMETER-KEY FILE BUILT BY WRITE-
   PARAMS AND STORES EACH KEY IN A 12 CHARACTER BUFFER PRO-
   VIDED BY PROCEDURE UNITIALIZE. EXECUTED ONCE FOR EACH OF
   THE THIRTEEN KEYS.
*****************************************************************)

VAR J : INTEGER;
BEGIN
J := 0;
REPEAT
READ(PRMDECK,CH);
J := J + 1; BUFFER[J] := CH
UNTIL EOLN(PRMDECK);
READLN(PRMDECK)
END;
PROCEDURE INITIALIZE(VAR  PARAM:PARAMLOC);

(*****************************************************************
   THIS PROCEDURE RETRIEVES AND PASSES THE 12 CHARACTER KEY
   STORAGE BUFFER TO THE KEY READ ROUTINE (READPARAMS).
   CALLED FROM GETPARAMS.
*****************************************************************)

BEGIN
READPARAMS;
FOR I := 1 TO 12 DO
BEGIN
PARAM[I] := BUFFER[I]
END;
END;
PROCEDURE GETPARAMS;

(*****************************************************************
   CALLED FROM MAIN TO RETRIEVE THE PARAMETER KEYS BUILT AND
   STORED ON DISK AT THE BEGINNING OF THE JOB.
*****************************************************************)

BEGIN
WITH PARAMFILE DO BEGIN
INITIALIZE(TSC);
INITIALIZE(NTAC);
INITIALIZE(MTAC);
..
```

```
INITIALIZE(WRC);
INITIALIZE(LPC);
INITIALIZE(CRC);
INITIALIZE(EQC);
INITIALIZE(MTC);
INITIALIZE(ALC);
INITIALIZE(STC);
INITIALIZE(LBC);
INITIALIZE(RQC);
END;
END;
PROCEDURE SAVEDATA;

(***************************************************************
   WRITE THE VALID ACCOUNTING RECORD TO THE OUTPUT FILE.
   THIS ROUTINE IS CALLED AFTER A VALID KEY HAS BEEN DETECTED.
   CALLED FROM PROCEDURE GETLINE.
***************************************************************)

BEGIN
WRITE(REDUCED,SPACE);
FOR I := 2 TO 72 DO
WRITE(REDUCED,DLINE[I]);
WRITELN(REDUCED)
END;

(***************************************************************
   THE FOLLOWING SET OF COMPARE LOCATION FUNCTIONS ARE INVOKED
   BY THE GETLINE PROCEDURE TO MAKE A CHARACTER-BY-CHARACTER
   SEARCH OF THE INPUT RECORD FOR A MESSAGE MATCHING THAT OF A
   SPECIFIC KEY. THE KEYS ARE PASSED TO THE FUNCTIONS THROUGH
   THE LOCAL VARIABLE, CLOC, WHILE THE INPUT DATA RECORD IS
   PASSED THROUGH THE GLOBAL VARIABLE, DLINE.
***************************************************************)


FUNCTION COMPLOC4 (VAR CLOC : PARAMLOC):BOOLEAN;

(****************************************************
   TEST THE CURRENT RECORD FOR SEVEN OR NINE TRACK
   TAPE ASSIGNMENT
****************************************************)

BEGIN
J:=0; KEY:=0;
FOR I := 20 TO 23 DO BEGIN
J := J + 1;
IF DLINE[I] = CLOC[J] THEN
KEY := KEY + 1
END;
IF KEY = 4 THEN
COMPLOC4 := TRUE
..
```

```
ELSE COMPLOC4 := FALSE
END;

(****************************************************
   SEE IF THE CURRENT INPUT RECORD IS AN ARRIVING
   JOB OR LINES PRINTED STATISTIC, RETURN TRUE
   IF YES
****************************************************)

FUNCTION COMPLOC7 (VAR CLOC : PARAMLOC):BOOLEAN;
BEGIN
J:=0; KEY:=0;
FOR I := 32 TO 38 DO BEGIN
J := J + 1;
IF DLINE[I] = CLOC[J] THEN
KEY := KEY + 1
END;
IF KEY = 7 THEN
COMPLOC7 := TRUE
ELSE COMPLOC7 := FALSE
END;

(****************************************************
   RETURN TRUE IF CURRENT RECORD DENOTES TAPE
   BLOCKS READ OR WRITTEN
****************************************************)

FUNCTION COMPLOC5 (VAR CLOC:PARAMLOC):BOOLEAN;
BEGIN
J:=0; KEY:=0;
FOR I := 26 TO 33 DO BEGIN
J := J + 1;
IF DLINE[I] = CLOC[J] THEN
KEY := KEY + 1
END;
IF KEY = 8 THEN
COMPLOC5 := TRUE
ELSE COMPLOC5 := FALSE
END;

(****************************************************
   TESTING FOR  (1) DISK MOUNTING
                (2) DISK MOUNT ACKNOWLEDGE
                (3) REQUEST FOR TAPE DRIVE
                (4) START OF SORT
   RETURN YES TO SENDING ROUTINE
****************************************************)

FUNCTION COMPLOC6 (VAR CLOC:PARAMLOC):BOOLEAN;
BEGIN
..
```

```
J:=0; KEY:=0;
FOR I := 20 TO 25 DO BEGIN
J := J + 1;
IF DLINE[I] = CLOC[J] THEN
KEY := KEY + 1
END;
IF KEY = 6 THEN
COMPLOC6 := TRUE
ELSE COMPLOC6 := FALSE
END;

(****************************************************
   RETURN TRUE IF CURRENT RECORD DENOTES A JOB
   ARRIVAL TO THE INPUT QUEUE
****************************************************)

FUNCTION COMPLOC12 (VAR CLOC:PARAMLOC):BOOLEAN;
BEGIN
J:=0; KEY:=0;
FOR I := 20 TO 31 DO BEGIN
J := J + 1;
IF DLINE[I] = CLOC[J] THEN
KEY := KEY + 1
END;
IF KEY = 12 THEN
COMPLOC12 := TRUE
ELSE COMPLOC12 := FALSE
END;
FUNCTION TAPE:BOOLEAN;

(****************************************************
   RETURN TRUE IF THE REQUEST RECORD CONTAINS *
   INDICATING A REQUEST FOR DISK SPACE INSTEAD
   OF A TAPE FILE
****************************************************)

BEGIN
TAPE := TRUE;
FOR I := 31 TO 37 DO
IF DLINE[I] = '*' THEN
TAPE := FALSE
END;



(*******************************************************
   THIS PROCEDURE GETS THE NEW LINE OF DATA FROM
   THE INPUT DATA FILE FOR USE BY THE GETLINE PROCEDURE
*******************************************************)

PROCEDURE GETDFDATA;
BEGIN
I := 0;
FOR J := 1 TO MAXD DO
```

```
DLINE[J] := SPACE;
IF NOT EOLN THEN
BEGIN
FOR J := 1 TO 4 DO
READ(CH);
END;
WHILE NOT EOLN DO
BEGIN
I := I + 1;
READ(CH);
IF I < 100 THEN
DLINE[I] := CH
END;
READLN
END;
PROCEDURE GETLINE;

(*******************************************
   THIS PROCEDURE CHECKS THE KEY PARAMETER FIELDS
   TO DETERMINE WHICH OF THE DATA COMPARE SUBROUTINES
   TO ENTER IN DETERMINING TYPE OF RECORD READ
**************************************************)

BEGIN
I := 0;
J := 0;
IF DLINE[20] = '$' THEN
BEGIN
IF DLINE[21] <> 'R' THEN SAVEDATA
END
ELSE BEGIN
WITH PARAMFILE DO
BEGIN
IF COMPLOC4(NTAC) = TRUE THEN SAVEDATA
ELSE IF COMPLOC4(MTAC) = TRUE THEN SAVEDATA
ELSE IF COMPLOC7(LPC) = TRUE THEN SAVEDATA
ELSE
BEGIN
IF COMPLOC7(CRC) = TRUE THEN SAVEDATA
ELSE IF COMPLOC6(MTC) = TRUE THEN SAVEDATA
ELSE IF COMPLOC6(ALC) = TRUE THEN SAVEDATA
ELSE IF COMPLOC6(STC) = TRUE THEN SAVEDATA
ELSE BEGIN
IF COMPLOC6(LBC) = TRUE THEN SAVEDATA
ELSE IF COMPLOC6(RQC) = TRUE THEN
BEGIN
IF TAPE = TRUE THEN SAVEDATA
END
ELSE BEGIN
IF COMPLOC12(TSC) = TRUE THEN SAVEDATA
ELSE IF COMPLOC12(EQC) = TRUE THEN SAVEDATA
ELSE IF COMPLOC5(WRC) = TRUE THEN SAVEDATA
END
END
END
END
END
..
```

```
END;
BEGIN (*MAIN*)
REWRITE(PR4DECK);
WRITEPARAMS; REWRITE(REDUCED);
RESET(PR4DECK);
GETPARAMS;
WHILE NOT EOF DO BEGIN
GETOFDATA;
GETLINE
END
END. (*MAIN*)
..
```

APPENDIX B

SPSS ROUTINES

```
.PROC,JULY
BON91,T2400,CM120000,154000,KT1.  T800710 BONDURANT AFIT BOX 4572
LIMIT,3200.
COMMENT. REDUCTION OF WEEKLY DAYFILE TAPE
COMMENT. INPUT IS TAPE1, ANY WEEKLY FORMAT DF TAPE
COMMENT. FINAL OUTPUT IS A ROUTED LISTING
COMMENT. OF SPSS CONDESCRIPTIVE AND MULTIPLE
COMMENT. REGRESSION OF THE WEEKLY ACTIVITY
COMMENT. FILE CATALOGUED IS PRMFILE USING 6 RBS
COMMENT. JOB CAN ONLY RUN AFTER 1400 HRS
VSN,TAPE1=X00188=C06515.
REQUEST,TAPE1,NT,E,S.
REQUEST,DFDATA,*PF.
REQUEST,REDDF,*PF.
COPYBF,TAPE1,DFDATA.
CATALOG,DFDATA,RP=999.
REWIND,DFDATA.
ATTACH,PASCLIB,ID=AFIT.
LIBRARY,PASCLIB.
RFL,120000.
ATTACH,LGO,DFRDWB,ID=T800710,MR=1.
REWIND,LGO,PASCLIB,DFDATA.
LGO,DFDATA,REDDF,DF.
CATALOG,REDDF,ID=T800710,RP=999.
PURGE,DFDATA.
RETURN,LGO,DF.
REWIND,REDDF.
UNLOAD,TAPE1.
REQUEST,SRTEDDF,*PF.
ATTACH,SORTF,ID=T800710,MR=1.
RFL,120000.
FILE(SRTEDDF,BT=C,FO=SQ,MRL=100,RT=Z)
FILE(REDDF,BT=C,FO=SQ,MRL=100,RT=Z)
REWIND,REDDF,SRTEDDF,SORTF.
LDSET(FILES=REDDF/SRTEDDF)
SORTMRG(I=SORTF/R)
CATALOG,SRTEDDF,ID=T800710,RP=999.
PURGE,REDDF.
REQUEST,PRMFILE,*PF.
ATTACH,LGO,PRAMXB,ID=T800710,MR=1.
REWIND,LGO,PASCLIB,SRTEDDF.
..
```

```
LGO,SRTEDDF,PRMFILE,DF.
RETURN,LGO,DF.
CATALOG,PRMFILE,ID=T800710,RP=999.
PURGE,SRTEDDF.
ATTACH,SPSS,ID=AFIT.
ATTACH,DSPSST,ID=T800710,MR=1.
REWIND,PRMFILE,SPSS,DSPSST.
SPSS,I=DSPSST,D=PRMFILE,L=LIST,NR.
REWIND,OUTPUT.
REWIND,LIST.
COPYSBF,LIST,OUTPUT.
RETURN,SPSS,LGO,DSPSST.

ROUTE,OUTPUT,DC=PR,TID=91,FID=BON,ST=CSB.
EXIT(S)
CATALOG,REDDF,ID=T800710,RP=999.
RETURN,LGO,DF.
REWIND,REDDF.
UNLOAD,TAPE1.
PURGE,DFDATA.
REQUEST,SRTEDDF,*PF.
ATTACH,SORTF,ID=T800710,MR=1.
RFL,120000.
FILE(SRTEDDF,BT=C,FO=SQ,MRL=100,RT=Z)
FILE(REDDF,BT=C,FO=SQ,MRL=100,RT=Z)
REWIND,REDDF,SRTEDDF,SORTF.
LDSET(FILES=REDDF/SRTEDDF)
SORTMRG(I=SORTF/R)
CATALOG,SRTEDDF,ID=T800710,RP=999.
PURGE,REDDF.
REQUEST,PRMFILE,*PF.
ATTACH,LGO,PRAMXB,ID=T800710,MR=1.
REWIND,LGO,PASCLIB,SRTEDDF.
LGO,SRTEDDF,PRMFILE,DF.
CATALOG,PRMFILE,ID=T800710,RP=999.
PURGE,SRTEDDF.
RETURN,LGO,DF.
ATTACH,SPSS,ID=AFIT.
ATTACH,DSPSST,ID=T800710,MR=1.
REWIND,PRMFILE,SPSS,DSPSST.
SPSS,I=DSPSST,D=PRMFILE,L=LIST,NR.
REWIND,OUTPUT.
REWIND,LIST.
COPYSBF,LIST,OUTPUT.
RETURN,SPSS,LGO,DSPSST.

ROUTE,OUTPUT,DC=PR,TID=91,FID=BON,ST=CSB.
*EOR
```

```
*****************************************************
*       THIS ROUTINE RECEIVES THE RAW PARAMETERIZED
*       JOB CASES CREATED BY THE EXTRACTION PROGRAM,    *
*       PRAMEX, CONVERTS THE TIME VARIABLES TO MINUTES *
*       AND THE UNITS VARIABLES TO THOUSANDS WHERE APPLI*
*       CABLE                                            *
*****************************************************

RUN NAME        CYBER THRUPUT ANALYSIS          ,
TASK NAME       DSPSS INITIAL SYSTEM FILE BUILD
FILE NAME       PRMFILE
VARIABLE LIST   JOB,DAY,ARRIVAL,DISKWAIT,THRUPUT,SORT,REQ7,REQ9,TREQ,
                CMREQ,DISKIO,CORE,MAXMEM,TAPEWAIT,PPUSAGE,SS,CPU,
                CM,IO,LINES,SORTED,TAPEIO,SEVENTRK,NINETRK,
                POSTPROC,INQUEUE,TURNAROUND
INPUT FORMAT    FIXED(A8,1X,F2.0,4(1X,F7.4),2(1X,F2.0),
                1X,F4.0,4(1X,F6.0),1X,F7.4/1X,
                F10.3,F10.2,1X,F8.2,2(1X,F8.1),
                1X,F7.0,1X,F6.0,1X,F7.0,2(1X,F2.0),3(1X,F7.4))
MISSING VALUES  DAY TO SORT DISKIO TO TAPEIO POSTPROC TO TURNAROUND(0.0)/
                REQ7 TO DISKIO SEVENTRK NINETRK(0)J
VAR LABELS      DAY DAY OF MONTH/
                ARRIVAL TIME OF JOB ARRIVAL/
                DISKWAIT MINUTES WAITING DISK MOUNTING/
                TURNAROUND TURNAROUND TIME IN MINUTES/
                SORT MINUTES REQUIRED TO SORT/
                REQ7 NUMBER OF 7 TRACK TAPES REQUESTED/
                REQ9 NUMBER OF 9 TRACK TAPES REQUESTED/
                TREQ TIME REQUESTED ON JOB CARD    000/
                CMREQ CM WORDS REQUESTED   000/
                DISKIO NUMBER OF DISK ACCESSES    000/
                CORE AMOUNT OF CORE USED   000/
                MAXMEM MAXIMUM CORE FOR ALL JOB STEPS    000/
                TAPEWAIT MINUTES WAITING TAPE MOUNTING/
                PPUSAGE TIME USING PPS  MINUTES/
                SS TIME IN SYSTEM SECONDS
                CPU TIME IN CPU  MINUTES/
                CM TIME IN CENTRAL MEMORY  MINUTES/
                IO TIME FOR ALL IO FUNCTIONS  MINUTES/
                LINES TOTAL LINES PRINTED   000/
                SORTED RECORDS SORTED DURING JOB   000/
```

```
                  TAPEIO NUMBER OF BLOCKS WRITTEN TO TAPE    000/
                  SEVENTRK NUMBER OF 7 TRACK TAPES USED/
                  NINETRK NUMBER OF 9 TRACK TAPES USED/
                  THRUPUT TIME FROM INQUEUE TO POSTPROC/
                  INQUEUE TIME IN INPUT QUEUE IN MINUTES/
                  POSTPROC TIME IN OUTPUT QUEUE THRU PRINT
COMPUTE           SS = SS / 60.0
IF                (DISKWAIT LT 0) DISKWAIT = 0.0
IF                (TAPEWAIT LT 0) TAPEWAIT = 0.0
IF                (POSTPROC LT 0)POSTPROC=0.0
IF                (POSTPROC GT 0)POSTPROC=POSTPROC * 60.0
IF                (INQUEUE LT 0)INQUEUE=0.0
IF                (TAPEIO GT 0) TAPEIO = TAPEIO/1000.0
IF                (CMREQ GT 0) CMREQ = CMREQ/1000.0
IF                (DISKIO GT 0) DISKIO = DISKIO/1000.0
IF                (IO GT 0) IO = IO/60.0
COMPUTE           INQUEUE = INQUEUE * 60.0
IF                (THRUPUT LT 0) THRUPUT = 0.0
IF                (SS LT 0) SS = 0.0
IF                (SORT LT 0) SORT = 0.0
IF                (TURNAROUND GT 0)TURNAROUND = TURNAROUND * 60.0
IF                (THRUPUT GT 0) THRUPUT = THRUPUT * 60.0
IF                (TAPEWAIT GT 0) TAPEWAIT = TAPEWAIT * 60.0
IF                (TAPEWAIT GT 1440)TAPEWAIT = 0.0
IF                (REQ7 GT 9)REQ7 = 0
IF                (REQ9 GT 7)REQ9 = 0
IF                (TURNAROUND LT 0)TURNAROUND = THRUPUT+INQUEUE+POSTPROC
IF                (SORTED GT 0) SORTED = SORTED/10.0
IF                (SORT GT 0) SORT = SORT/60.0
IF                (DISKWAIT GT 0)DISKWAIT=DISKWAIT*60.0
IF                (SEVENTRK GT 9) SEVENTRK = REQ7
IF                (NINETRK GT 7) NINETRK = REQ9
COMPUTE           TREQ = TREQ / 60.0
COMPUTE           CORE = CORE/1000.0
COMPUTE           MAXMEM = MAXMEM/100.0
COMPUTE           PPUSAGE = PPUSAGE/60.0
COMPUTE           CPU = CPU/60.0
COMPUTE           CM = CM/60.0
COMPUTE           IARRIVAL = ARRIVAL
COMPUTE           LINES = LINES/1000.0
COMPUTE           IOWAIT = TAPEWAIT + DISKWAIT
IF                (SS GT 0)SWAPROLL=T
                                      HRUPUT-SS
IF                (SWAPROLL LT 0)SWAPROLL=0.0
IF                (CM GT (CPU+IO))EXQUEUE=CM-(CPU+IO)
COMPUTE           SWAPTIME=THRUPUT - (CPU+IO+IOWAIT+EXQUEUE)
IF                (SWAPTIME LT 0)SWAPTIME=0.0
COMPUTE           WAITTIME=IOWAIT + SWAPTIME
COMPUTE           QUETIME=INQUEUE+POSTPROC
COMPUTE           EXECUTN=CPU+CM+IO+EXQUEUE
INPUT MEDIUM      DISK
..
```

```
N OF CASES      UNKNOWN
READ INPUT DATA
SORT CASES      DAY
SAVE FILE       PRMFILE
FINISH

*****************************************************
*      THIS ROUTINE STRATIFIES THE DATA ACCORDING TO  *
*      THE DAYS OF THE MONTH PROCESSED. DAYS FROM THE *
*      THE MONTHS OF JUNE AND AUGUST ARE FIRST GROUPED*
*      THEN DELETED FROM THE FINAL FILE SO THAT ONLY  *
*      JOBS RUN DURING THE MONTH OF JULY ARE REMAINING*
****************************************************

RUN NAME        CYBER THRUPUT ANALYSIS
TASK NAME       BREAKOUT BY DAY
FILE NAME       PRMFILE
GET FILE        PRMFILE
SUBFILE LIST    DA1(214),X1(185),DA2(309),X2(155),
                DA3(166),DA4(147),DA5(219),X3(1),DA6(221),
                DA7(318),DA8(267),DA9(305),DA10(297),DA11(195),
                DA12(120),X6(1),DA20(219),DA21(289),DA22(270),
                DA23(284),DA24(150),DA25(67),X7(1),DA26(64),
                DA27(137),DA28(194),X8(268),DA29(137),X9(283)
RUN SUBFILES    ALL
SAVE FILE       PRMFILE
FINISH
*EOR
RUN NAME        CYBER THRUPUT ANALYSIS
TASK NAME       DELETION OF EXCESS DAYS
FILE NAME       PRMFILE
GET FILE        PRMFILE
DELETE SUBFILES X1,X2,X3,X6,X7,X8,X9
SORT CASES      DAY,ARRIVAL
SAVE FILE       PRMFILE
FINISH
*EOR

*****************************************************
*      THIS RUN DETERMINES THE NUMBER OF JOBS RUN BY  *
*      DAY IN ORDER TO GROUP JOBS INTO SUBFILES ON THE*
*      NEXT RUN. THE NUMBER OF CASES PER DAY WILL     *
*      HAVE TO BE MANUALLY KEYED IN ON THE NEXT RUN   *
*      PRIOR TO ITS RUNNING. THUS, NO AUTOMATIC       *
*      CONTINUATION. IF THE DATA IS TO BE RUN AS ONE  *
*      COMPLETE POPULATION, AS WAS AULTIMATELY  DONE *
*      FOR THIS ANALYSIS, THIS ROUTINE CAN BY SKIPPED *
*****************************************************

RUN NAME        CYBER THRUPUT ANALYSIS
..
```

```
TASK NAME        FREQUENCY OF DATA BY DAY
FILE NAME        PRMFILE
GET FILE         PRMFILE
RUN SUBFILES     ALL
FREQUENCIES      INTEGER=DAY(1,31)
OPTIONS          5
FINISH
*EOR

*****************************************************
*     THIS ROUTINE CONVERTS THE JOB ARRIVAL  TIME    *
*     FORM SECONDS TO MINUTES. USING A LAG FUNCTION, *
*     THE ARRIVAL TIME OF THE PREVIOUS JOB IS SUB-   *
*     TRACTED FROM THE CURRENT JOB (THE JOBS WERE    *
*     SORTED BY DAY AND TIME OF ARRIVAL IN THE INIT  *
*     IAL RUN) PROVIDING THE TIME BETWEEN ARRIVAL.   *
*     A BY-PRODUCT, AN EFFECTIVE INTER-ARRIVAL TIME  *
*     IS PRODUCED BY USING THE MULTIPROGRAMMING RATE *
*     TO PROVIDE A STATISTIC SIMULATING A UNI-PROGRAM*
*     MING MACHINE. THIS STATISTIC IS USED TO TEST A *
*     NUMBER OF THEORIES PERTAINING TO THE MULTI-    *
*     PROGRAMMING CAPABILITIES OF THE COMPUTER       *
*****************************************************

RUN NAME         CYBER THRUPUT ANALYSIS
TASK NAME        DIRIVATION OF INTER ARRIVAL TIMES
FILE NAME        PRMFILE
GET FILE         PRMFILE
COMPUTE          IR=LAG(ARRIVAL)
COMPUTE          IARRIVAL=ARRIVAL-IR
COMPUTE          IARRIVAL = IARRIVAL * 60.0
COMPUTE          EFFIARIV=IARRIVAL * 4.28
RUN SUBFILES     EACH
SAVE FILE        PRMFILE
FINISH
*EOR

*****************************************************
*     THIS RUN PRODUCES CONTINUOUS DATA DISCRIPTIVE  *
*     STATISTICS FOR MEANS, VARIANCES,HIGH AND LOW  *
*     VALUES, KURTOSIS, ETC., USED IN MAKING THE     *
*     INITIAL ASSUMPTIONS  ABOUT THE DATA            *U
*****************************************************

RUN NAME         CYBER THRUPUT ANALYSIS
TASK NAME        CONDESCRIPTIVE LOOK AT TURNAROUND VARIABLES
GET FILE         PRMFILE
RUN SUBFILES     ALL
CONDESCRIPTIVE ALL
CONDESCRIPTIVE ALL
..
```

```
OPTIONS         1
FINISH
*EOR

******************************************************
*      THIS ROUTINE IS USED TO PRODUCE MULTIPLE REGRES*
*      SION STATISTICS FOR THE ANALYSIS OF THE VARIOUS*
*      TURNAROUND TIME MODELS DEVELOPED FOR THE STUDY.*
*      THE MODEL VARIABLES FOR EACH OF THE MODELS ARE *
*      REGRESSED AGAINST THEIR RESPECTIVE DEPENDENT   *
*      VARIABLE IN A STEPWISE MANER                   *
******************************************************

RUN NAME        CYBER THRUPUT ANALYSIS
TASK NAME       REGRESSION ANALYSIS OF MODEL VARIABLES
GET FILE        PRMFILE
RUN SUBFILES    ALL
REGRESSION      VARIABLES=DISKIO,IO,TAPEIO TO NINETRK/
                REGRESSION=IO WITH DISKIO,TAPEIO,SEVENTRK,NINETRK(1)/
OPTIONS         1
STATISTICS      ALL
REGRESSION      VARIABLES=INQUEUE,CPU,IO,POSTPROC,IOWAIT,TURNAROUND,SWAPTIME,E
EUE/
                REGRESSION=TURNAROUND WITH INQUEUE,CPU,IO,POSTPROC,
                SWAPTIME,IOWAIT,EXQUEUE(1)/
OPTIONS         1
STATISTICS      ALL
REGRESSION      VARIABLES=INQUEUE,THRUPUT,POSTPROC,TURNAROUND/
                REGRESSION=TURNAROUND WITH INQUEUE,THRUPUT,POSTPROC(1)/
OPTIONS         1
STATISTICS      ALL
REGRESSION      VARIABLES=THRUPUT,DISKIO,PPUSAGE,CPU TO IO,TAPEIO
                IOWAIT,SWAPTIME/
                REGRESSION=THRUPUT WITH DISKIO,PPUSAGE,CPU TO IO,
                TAPEIO,IOWAIT,SWAPTIME(1)/
OPTIONS         1
STATISTICS      ALL
REGRESSION      VARIABLES=REQ7 TO CMREQ,IO,QUETIME/
                REGRESSION=QUETIME WITH REQ7 TO CMREQ,IO(1)/
OPTIONS         1
STATISTICS      ALL
REGRESSION      VARIABLES=INQUEUE,REQ7 TO CMREQ/
                REGRESSION=INQUEUE WITH REQ7 TO CMREQ(1)/
OPTIONS         1
STATISTICS      ALL
REGRESSION      VARIABLES=THRUPUT,LINES,SEVENTRK TO POSTPROC/
                REGRESSION=POSTPROC WITH THRUPUT,LINES,SEVENTRK,NINETRK(1)/
OPTIONS         1
STATISTICS      ALL
REGRESSION      VARIABLES=DISKWAIT,TAPEWAIT,SWAPTIME,WAITTIME/
                REGRESSION=WAITTIME WITH DISKWAIT,TAPEWAIT,SWAPTIME(1)/
..
```

```
OPTIONS         1
STATISTICS      ALL
REGRESSION      VARIABLES=THRUPUT, TURNAROUND, QUETIME/
                REGRESSION=TURNAROUND WITH THRUPUT, QUETIME(1)/
OPTIONS         1
STATISTICS      ALL
REGRESSION      VARIABLES=CPU TO IO, EXQUEUE, EXECUTN/
                REGRESSION=EXECUTN WITH CPU TO IO, EXQUEUE(1)/
OPTIONS         1
STATISTICS      ALL
FINISH
*EOR

*****************************************************
*     THIS RUN IS USED TO PULL A RANDOM SAMPLE OF     *
*     ONE PERCENT OF THE DATA CASES AND REGRESS THE   *
*     RESULTS AGAINST THE TURNAROUND MODEL TO FURTHER*
*     TEST ITS ACCURACY IN PREDICTING THE TRUE TURN-  *
      AROUND TIME FOR JOBS                            *
*****************************************************

RUN NAME        CYBER THRUPUT ANALYSIS
TASK NAME       TEST OF DIFFERENCES BETWEEN WEEKLY PROCESSING
GET FILE        PRMFILE
RUN SUBFILES    ALL
RECODE          DAY(LOWEST THRU 5=1)
                (6 THRU 12=2)
                (13 THRU 19=3)
                (20 THRU 26=4)
                (27 THRU HIGHEST=5)(ELSE=0)
T-TEST          GROUPS=DAY(1,2)/VARIABLES=ARRIVAL, SS, CPU, IO, TURNAROUND,
                SORTED, SEVENTRK, NINETRK
T-TEST          GROUPS=DAY(1,3)/VARIABLES=ARRIVAL, SS, CPU, IO, TURNAROUND,
                SORTED, SEVENTRK, NINETRK
T-TEST          GROUPS=DAY(1,4)/VARIABLES=ARRIVAL, SS, CPU, IO, TURNAROUND,
                SORTED, SEVENTRK, NINETRK
T-TEST          GROUPS=DAY(1,5)/VARIABLES=ARRIVAL, SS, CPU, IO, TURNAROUND,
                SORTED, SEVENTRK, NINETRK
T-TEST          GROUPS=DAY(2,3)/VARIABLES=ARRIVAL, SS, CPU, IO, TURNAROUND,
                SORTED, SEVENTRK, NINETRK
T-TEST          GROUPS=DAY(2,4)/VARIABLES=ARRIVAL, SS, CPU, IO, TURNAROUND,
                SORTED, SEVENTRK, NINETRK
T-TEST          GROUPS=DAY(2,5)/VARIABLES=ARRIVAL, SS, CPU, IO, TURNAROUND,
                SORTED, SEVENTRK, NINETRK
T-TEST          GROUPS=DAY(3,4)/VARIABLES=ARRIVAL, SS, CPU, IO, TURNAROUND,
                SORTED, SEVENTRK, NINETRK
T-TEST          GROUPS=DAY(3,5)/VARIABLES=ARRIVAL, SS, CPU, IO, TURNAROUND,
                SORTED, SEVENTRK, NINETRK
T-TEST          GROUPS=DAY(4,5)/VARIABLES=ARRIVAL, SS, CPU, IO, TURNAROUND,
                SORTED, SEVENTRK, NINETRK
..
```

```
FINISH
*EOR
RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         ONE PERCENT SAMPLE OF ALL CASES
FILE NAME         PRMFILE
GET FILE          PRMFILE
SAMPLE            0.01
CONDESCRIPTIVE    ALL
OPTIONS           1
SAVE FILE         PRMFILE
FINISH
*EOR
RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         REGRESSION ANALYSIS OF ONE PERCENT DATA
GET FILE          PRMFILE
RUN SUBFILES      ALL
REGRESSION        VARIABLES=INQUEUE,CPU,IO,SWAPTIME,IOWAIT,POSTPROC,TURNAROUND/
                  REGRESSION=TURNAROUND WITH CPU,IO,INQUEUE,SWAPTIME,
                  IOWAIT,POSTPROC(1)/
FINISH
*EOR

*****************************************************
*     THIS RUN IS USED TO TEST A HYPOTHESIS THAT      *
*     VARIABLES OTHER THAN THOSE REQUESTED ON THE    *
*     JOB CARD HAS AN EFFECT ON THE AMOUNT  OF TIME   *
*     A JOB SPENDS IN THE INPUT QUEUE                 *
*****************************************************

RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         REGRESSION OF INQUEUE VARIABLES
GETFILE           PRMFILE
RUN SUBFILES      ALL
REGRESSION        VARIABLES=INQUEUE,CPU,IO,CORE,SEVENTRK,NINETRK,
                  TURNAROUND/
                  REGRESSION=INQUEUE WITH CPU,IO,CORE,SEVENTRK,NINETRK,
                  TURNAROUND(1)/
OPTIONS           1
FINISH
*EOR
RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         THRUPUT TIMES FOR JOBS W/NO IO RESOURCE VS IO RESOURCES
GET FILE          PRMFILE
*SELECT IF        (TAPEIO GT 0)
CONDESCRIPTIVE    THRUPUT
*SELECT IF        (NOT(TAPEIO GT 0))
CONDESCRIPTIVE    THRUPUT
CONDESCRIPTIVE    THRUPUT
OPTIONS           1
FINISH
..
```

```
*EOR

****************************************************
*      THIS ROUTINE IS USED TO STRATIFY JOBS BY WORK- *
*      SHIFT FOR TESTING A HYPOTHESIS THAT A DIFFER-  *
*      ENCE EXISTS IN THE NUMBER AND SIZE OF JOBS     *
*      PROCESSED BY THE DIFFERENT SHIFTS             *
****************************************************

RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         MEAN VALUES BY SHIFT
FILE NAME         PRMFILE
GET FILE          PRMFILE
SELECT IF         (ARRIVAL LE 08.0)
SELECT IF         (ARRIVAL GT 08.0 AND LE 16.0)
SELECT IF         (ARRIVAL GT 16.0)
SUBFILE LIST      S1(1311),S2(2378),S3(1375)
RUN SUBFILES      ALL
CONDESCRIPTIVE    CPU IO
T-TEST            GROUPS=S1,S2/VARIABLES=CPU,IO
T-TEST            GROUPS=S1,S3/VARIABLES=CPU,IO
T-TEST            GROUPS=S2,S3/VARIABLES=CPU,IO
SAVE FILE         PRMFILE
FINISH
*EOR

****************************************************
*      THIS ROUTINE IS USED TO CONVERT THE ALPHANUMER-*
*      IC JOB NAME; GROUP JOBS BY SYSTEM; AND SINGLE  *
*      OUT THOSE SYSTEMS FOUND IN THE UPPER QUARTILE  *
*      OF TURNAROUND TIME USAGE FOR THE PURPOSE OF    *
*      TESTING A THEORY ABOUT WHEN RESOURCES WOULD BE *
*      EXHAUSTED GIVEN THE CONTINUAL ADDITION OF NEW  *
*      SYSTEMS ON THE COMPUTER                        *
****************************************************

RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         INITIAL AGGREGATE FILE BUILD
FILE NAME         PRMFILE
VARIABLE LIST     W,O,R,K,U,N,T,TURNAROUND
INPUT FORMAT      FIXED(1X,7(A1)/89X,F7.4)
MISSING VALUES    TURNAROUND(0.0)
IF                (TURNAROUND GT 0)TURNAROUND = TURNAROUND * 60.0
IF                (TURNAROUND LT 0)TURNAROUND = 0.0
RECODE            W TO T('A'=20)('B'=21)('C'=22)('D'=23)('E'=24)
                  ('F'=25)('G'=26)('H'=27)('I'=28)('J'=29)('K'=30)
                  ('L'=31)('M'=32)('N'=33)('O'=34)('P'=35)('Q'=26)
                  ('R'=37)('S'=38)('T'=39)('U'=40)('V'=41)('W'=42)
                  ('X'=43)('Y'=44)('Z'=45)('.'=46)(CONVERT)
COMPUTE           WORKUNIT=W+O+R+K+U+N+T
..
```

```
INPUT MEDIUM      DISK
N OF CASES        UNKNOWN
READ INPUT DATA
SORT CASES        WORKUNIT
SAVE FILE         PRMFILE
FINISH
*EOR
RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         AGGREGATION OF WORKUNITS
FILE NAME         BCDOUT
GET FILE          PRMFILE
RUN SUBFILES      ALL
AGGREGATE         GROUPVARS=WORKUNIT/VARIABLES=TURNAROUND/
                  AGGSTATS=VALIDN, SUM,MEAN/
CONDESCRIPTIVE    TURNAROUND
SAVE FILE         BCDOUT
FINISH
*EOR
RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         MEAN VALUES OF SYSTEM TURNAROUND TIMES
VARIABLE LIST     WORKUNIT, TURNAROUND, VALIDN, SUM,MEAN
INPUT MEDIUM      DISK
INPUT FORMAT      BINARY
CONDESCRIPTIVE    WORKUNIT TO MEAN
FINISH
*EOR
..
```

```
RUN NAME         CYBER THRUPUT ANALYSIS
TASK NAME        HISTOGRAM RUN OF SUBFILES
GET FILE         PRIFILE
VALUE LABELS     ARRIVAL (1)0000-0200 (2)0200-0400 (3)0400-0600
                 (4)0600-0800 (5)0800-1000 (6)1000-NOON (7)NOON-1400
                 (8)1400-1600 (9)1600-1800 (10)1800-2000
                 (11)2000-2200 (12)2200-MID/
                 DISKWAIT (0)NONE (1)LESS THAN 1 MINUTE (2)1-2 MIN
                        (3)2-3 MIN (4)3-4 MIN (5)4-5 MIN
                        (6)5-6 MIN (7)6-7 MIN (8)7-8 MIN
                        (9)8-9 MIN (10)9 OR MORE/
                 SEVENTRK (0)TAPES/NINETRK (0)TAPES/
                 REQ7 (0)TAPES/REQ9 (0)TAPES/

RECODE           ARRIVAL (LOWEST THRU 2.0 = 1)
                        (2.001 THRU 4.0=2)
                        (4.001 THRU 6.0=3)
                        (6.001 THRU 8.0=4)
                        (8.001 THRU 10.0=5)
                        (10.001 THRU 12.0=6)
                        (12.001 THRU 14.0=7)
                        (14.001 THRU 16.0=8)
                        (16.001 THRU 18.0=9)
                        (18.001 THRU 20.0=10)
                        (20.001 THRU 22.0=11)
                        (22.001 THRU HIGHEST = 12)(ELSE=0)/
                 DISKWAIT(LOWEST THRU 0.01 = 0)
                        (0.011 THRU 1.0 = 1)
                        (1.001 THRU 2.0 = 2)
                        (2.001 THRU 3.0 = 3)
                        (3.001 THRU 4.0 = 4)
                        (4.001 THRU 5.0 = 5)
                        (5.001 THRU 6.0 = 6)
                        (6.001 THRU 7.0 = 7)
                        (7.001 THRU 8.0 = 8)
                        (8.001 THRU 9.0 = 9)
                        (9.001 THRU HIGHEST = 10)
RUN SUBFILES     ALL
FREQUENCIES      INTEGER=ARRIVAL(0,12) DISKWAIT(0,10)
OPTIONS          3,6,8
STATISTICS       ALL
..
```

```
FREQUENCIES     INTEGER=SEVENTRK(0,9) NINETRK(0,7)
OPTIONS         3,6,8
STATISTICS      ALL
FREQUENCIES     INTEGER=REQ7(0,9)
OPTIONS         3,6,8
STATISTICS      ALL
FREQUENCIES     INTEGER=REQ9(0,7)
OPTIONS         3,6,8
STATISTICS      ALL
FINISH
*EOR
RUN NAME        CYBER THRUPUT ANALYSIS
TASK NAME       HISTOGRAM RUN OF SUBFILES
GET FILE        PRIFILE
VALUE LABELS    ARRIVAL (1)0000-0200 (2)0200-0400 (3)0400-0600
                (4)0600-0800 (5)0800-1000 (6)1000-NOON (7)NOON-1400
                (8)1400-1600 (9)1600-1800 (10)1800-2000
                (11)2000-2200 (12)2200-MID/
                DISKWAIT (0)NONE (1)LESS THAN 1 MINUTE (2)1-2 MIN
                         (3)2-3 MIN (4)3-4 MIN (5)4-5 MIN
                         (6)5-6 MIN (7)6-7 MIN (8)7-8 MIN
                         (9)8-9 MIN (10)9 OR MORE/
                SEVENTRK (0)TAPES/NINETRK (0)TAPES/
                REQ7 (0)TAPES/REQ9 (0)TAPES/
                TURNAROUND (1)LESS THAN 1 MINUTE (2)1-5 MIN (3)5-10 MIN
                         (4)10-20 MIN (5)20-30 MIN (6)30-60 MIN
                         (7)60-90 MIN (8)90-120 MIN (9)2-3 HRS
                         (10)3-6 HRS (11)MORE THAN 6/
                IO (0)NONE (1)LESS THAN 1 MINUTE (2)1-5 MIN
                         (3)5-10 MIN (4)10-30 MIN (5)30-60 MIN
                         (6)1-2 HR (7)2-3 HR (8)3-6 HR (9)MORE THAN 6/
                SORTED (0)NONE (1)LESS THAN 1K (2)1-5K
                         (3)5-10K (4)10-30K (5)30-60K (6)60-120K
                         (7)120-180K (8)180-300K (9)MORE THAN 300K/
                CM (1)LESS THAN 1 MINUTE (2)1-5 MIN (3)5-10 MIN
                         (4)10-20 MIN (5)20-40 MIN (6)40-60 MIN
                         (7)60-90 MIN (8)90-120 MIN (9)MORE THAN 2 HRS/
                SORT (0)NONE (1)LESS THAN 1 MINUTE (2)1-5 MIN
                         (3)5-10 MIN (4)10-20 MIN (5)20-30 MIN
                         (6)30-60 MIN (7)60-90 MIN
                         (8)90-120 MIN (9)MORE THAN 2 HRS/
                DISKIO (1)LESS THAN 1K (2)1-5K (3)5-10K
                         (4)10-20K (5)20-40K (6)40-60K (7)MORE THAN 60K/
                CORE (1)LESS THAN 10K (2)10-50K (3)50-100K
                         (4)100-150K (5)150-200K (6)200-250K
                         (7)MORE THAN 250K/
                PPUSAGE (1)LESS THAN 1 MINUTE (2)1-5 MIN
                         (3)5-10 MIN (4)10-15 MIN (5)15-20 MIN
                         (6)20-30 MIN (7)30-60 MIN (8)60-90 MIN
                         (9)MORE THAN 90/
```

```
                  SNAPFILE (0)NONE (1)LESS THAN 1 MINUTE (2)1-5 MIN
                          (3)5-10 MIN (4)10-15 MIN (5)15-20 MIN
                          (6)20-30 MIN (7)30-60 MIN (8)60-90 MIN (9)MORE THA
N 90/
                  CPU (1)LESS THAN 6 SEC (2)6-12 SEC (3)12-24 SEC
                          (4)24-36 SEC (5)30-48 SEC (6)48-60 SEC
                          (7)1-5 MIN (8)5-10 MIN (9)10-20 MIN (10)MORE THAN
20/
                  TAPEIO (0)NONE (1)LESS THAN 1K (2)1-5K (3)5-10K
                          (4)10-20K (5)20-50K (6)50-100K
                          (7)100-150K (8)150-200K (9)MORE THAN 200K/

RECODE            TURNAROUND(LOWEST THRU 1.0 = 1)
                          (1.001 THRU 5.0 = 2)
                          (5.001 THRU 10.0 = 3)
                          (10.001 THRU 20.0 = 4)
                          (20.001 THRU 30.0 = 5)
                          (30.001 THRU 60.0 = 6)
                          (60.001 THRU 90.0 = 7)
                          (90.001 THRU 120.0 = 8)
                          (120.001 THRU 180.0 = 9)
                          (180.001 THRU 300.0 = 10)
                          (300.001 THRU HIGHEST = 11)(ELSE=0)
RUN SUBFILES      ALL
FREQUENCIES       INTEGER=TURNAROUND(0,11)
OPTIONS           3,6,8
STATISTICS        ALL
FINISH
*EOR
RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         HISTOGRAM RUN OF SUBFILES
GET FILE          PRMFILE
VALUE LABELS      IO (0)NONE (1)LESS THAN 1 MINUTE (2)1-5 MIN
                          (3)5-10 MIN (4)10-30 MIN (5)30-60 MIN
                          (6)1-2 HR (7)2-3 HR (8)3-6 HR (9)MORE THAN 6/
                  SORTED (0)NONE (1)LESS THAN 1K (2)1-5K
                          (3)5-10K (4)10-30K (5)30-60K (6)60-120K
                          (7)120-180K (8)180-300K (9)MORE THAN 300K/
RECODE            IO SORTED (LOWEST THRU 0.01 = 0)
                          (0.011 THRU 1.0 = 1)
                          (1.001 THRU 5.0 = 2)
                          (5.001 THRU 10.0 = 3)
                          (10.001 THRU 30.0 = 4)
                          (30.001 THRU 60.0 = 5)
                          (60.001 THRU 120.0 = 6)
                          (120.001 THRU 180.0 = 7)
                          (180.001 THRU 300.0 = 8)
                          (300.1 THRU HIGHEST = 9)
RUN SUBFILES      ALL
FREQUENCIES       INTEGER=IO SORTED(0,9)
OPTIONS           3,6,8
STATISTICS        ALL
..
```

```
FINISH
*EOR
RUN NAME         CYBER THRUPUT ANALYSIS
TASK NAME        HISTOGRAM RUN OF SUBFILES
GET FILE         PRMFILE
VALUE LABELS     C4 (1)LESS THAN 1 MINUTE (2)1-5 MINUTES (3)5-10 MIN
                         (4)10-20 MIN (5)20-40 MIN (6)40-60 MIN
                         (7)60-90 MIN (8)90-120 MIN (9)MORE THAN 2 HRS/
                 SORT (0)NONE (1)LESS THAN 1 MINUTE (2)1-5 MIN
                         (3)5-10 MIN (4)10-20 MIN (5)20-30 MIN
                         (6)30-60 MIN (7)60-90 MIN
                         (8)90-120 MIN (9)MORE THAN 2 HRS/
RECODE           C4(LOWEST THRU 1.0 = 1)
                         (1.001 THRU 5.0 = 2)
                         (5.001 THRU 10.0 = 3)
                         (10.001 THRU 20.0 = 4)
                         (20.001 THRU 40.0 = 5)
                         (40.001 THRU 60.0 = 6)
                         (60.001 THRU 90.0 = 7)
                         (90.001 THRU 120.0 = 8)
                         (120.001 THRU HIGHEST = 9)(ELSE=0)/
                 SORT(LOWEST THRU 0.001 = 0)
                         (0.002 THRU 1.0 = 1)
                         (1.001 THRU 5.0 = 2)
                         (5.001 THRU 10.0 = 3)
                         (10.001 THRU 20.0 = 4)
                         (20.001 THRU 30.0 = 5)
                         (30.001 THRU 60.0 = 6)
                         (60.001 THRU 90.0 = 7)
                         (90.001 THRU 120.0 = 8)
                         (120.001 THRU HIGHEST = 9)
RUN SUBFILES     ALL
FREQUENCIES      INTEGER=C4 SORT(0,9)
OPTIONS          3,6,8
STATISTICS       ALL
FINISH
*EOR
RUN NAME         CYBER THRUPUT ANALYSIS
TASK NAME        HISTOGRAM RUN OF SUBFILE
GET FILE         PRMFILE
VALUE LABELS     ARRIVAL (1)0000-0200 (2)0200-0400 (3)0400-0600
                 (4)0600-0800 (5)0800-1000 (6)1000-NOON (7)NOON-1400
                 (8)1400-1600 (9)1600-1800 (10)1800-2000
                 (11)2000-2200 (12)2200-MID/
                 DISKWAIT (0)NONE (1)LESS THAN 1 MINUTE (2)1-2 MIN
                         (3)2-3 MIN (4)3-4 MIN (5)4-5 MIN
                         (6)5-6 MIN (7)6-7 MIN (8)7-8 MIN
                         (9)8-9 MIN (10)9 OR MORE/
                 SEVENTRK (0)TAPES/NINETRK (0)TAPES/
                 REQ7 (0)TAPES/REQ9 (0)TAPES/
```

```
TURNAROUND (1)LESS THAN 1 MINUTE (2)1-5 MIN (3)5-10 MIN
        (4)10-20 MIN (5)20-30 MIN (6)30-60 MIN
        (7)60-90 MIN (8)90-120 MIN (9)2-3 HRS
        (10)3-6 HRS (11)MORE THAN 6/
IO (0)NONE (1)LESS THAN 1 MINUTE (2)1-5 MIN
        (3)5-10 MIN (4)10-30 MIN (5)30-60 MIN
        (6)1-2 HR (7)2-3 HR (8)3-6 HR (9)MORE THAN 6/
SORTED (0)NONE (1)LESS THAN 1K (2)1-5K
        (3)5-10K (4)10-30K (5)30-60K (6)60-120K
        (7)120-180K (8)180-300K (9)MORE THAN 300K/
CT (1)LESS THAN 1 MINUTE (2)1-5 MIN (3)5-10 MIN
        (4)10-20 MIN (5)20-40 MIN (6)40-60 MIN
        (7)60-90 MIN (8)90-120 MIN (9)MORE THAN 2 HRS/
SORT (0)NONE (1)LESS THAN 1 MINUTE (2)1-5 MIN
        (3)5-10 MIN (4)10-20 MIN (5)20-30 MIN
        (6)30-60 MIN (7)60-90 MIN
        (8)90-120 MIN (9)MORE THAN 2 HRS/
DISKIO (1)LESS THAN 1K (2)1-5K (3)5-10K
        (4)10-20K (5)20-40K (6)40-60K (7)MORE THAN 60K/
CORE (1)LESS THAN 10K (2)10-50K (3)50-100K
        (4)100-150K (5)150-200K (6)200-250K
        (7)MORE THAN 250K/
PPUSAGE (1)LESS THAN 1 MINUTE (2)1-5 MIN
        (3)5-10 MIN (4)10-15 MIN (5)15-20 MIN
        (6)20-30 MIN (7)30-60 MIN (8)60-90 MIN
        (9)MORE THAN 90/
SWAPTIME (0)NONE (1)LESS THAN 1 MINUTE (2)1-5 MIN
        (3)5-10 MIN (4)10-15 MIN (5)15-20 MIN
        (6)20-30 MIN (7)30-60 MIN (8)60-90 MIN (9)MORE THA
N 90/
CPU (1)LESS THAN 6 SEC (2)6-12 SEC (3)12-24 SEC
        (4)24-36 SEC (5)36-48 SEC (6)48-60 SEC
        (7)1-5 MIN (8)5-10 MIN (9)10-20 MIN (10)MORE THAN
20/
TAPEIO (0)NONE (1)LESS THAN 1K (2)1-5K (3)5-10K
        (4)10-20K (5)20-50K (6)50-100K
        (7)100-150K (8)150-200K (9)MORE THAN 200K/

RECODE      DISKIO(LOWEST THRU 1.0 = 1)
        (1.001 THRU 5.0 = 2)
        (5.001 THRU 10.0 = 3)
        (10.001 THRU 20.0 = 4)
        (20.001 THRU 40.0 = 5)
        (40.001 THRU 60.0 = 6)
        (60.001 THRU HIGHEST = 7)(ELSE=0)/
    CORE(LOWEST THRU 10.0 = 1)
        (10.1 THRU 50.0 = 2)
        (50.1 THRU 100.0 = 3)
        (100.0 THRU 150.0 = 4)
        (150.1 THRU 200.0 = 5)
        (200.1 THRU 250.0 = 6)
        (250.1 THRU HIGHEST = 7)(ELSE=0)
```

```
RUN SUBFILES      ALL
FREQUENCIES       INTEGER=DISKIO(0,9) CORE(0,7)
OPTIONS           3,6,8
STATISTICS        ALL
FINISH
*EOR
RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         HISTOGRAM RUN OF SUBFILES
GET FILE          PRAFILE
VALUE LABELS      PPUSAGE (1)LESS THAN 1 MINUTE (2)1-5 MIN
                          (3)5-10 MIN (4)10-15 MIN (5)15-20 MIN
                          (6)20-30 MIN (7)30-60 MIN (8)60-90 MIN
                          (9)MORE THAN 90/
                  SWAPTIME (0)NONE (1)LESS THAN 1 MINUTE (2)1-5 MIN
                          (3)5-10 MIN (4)10-15 MIN (5)15-20 MIN
                          (6)20-30 MIN (7)30-60 MIN (8)60-90 MIN (9)MORE THA
N 90/
                  CPU (1)LESS THAN 6 SEC (2)6-12 SEC (3)12-24 SEC
                          (4)24-36 SEC (5)36-48 SEC (6)48-60 SEC
                          (7)1-5 MIN (8)5-10 MIN (9)10-20 MIN (10)MORE THAN
20/
                  TAPEIO (0)NONE (1)LESS THAN 1K (2)1-5K (3)5-10K
                          (4)10-20K (5)20-50K (6)50-100K
                          (7)100-150K (8)150-200K (9)MORE THAN 200K/

RECODE            PPUSAGE (LOWEST THRU 1.0 =1)
                          (1.1 THRU 5.0 = 2)
                          (5.001 THRU 10.0 = 3)
                          (10.001 THRU 15.0 = 4)
                          (15.001 THRU 20.0 = 5)
                          (20.001 THRU 30.0 = 6)
                          (30.001 THRU 60.0 = 7)
                          (60.001 THRU 90.0 = 8)
                          (90.001 THRU HIGHEST = 9)(ELSE=0)/
                  SWAPTIME(LOWEST THRU 0.01 = 0)
                          (0.011 THRU 1.0 = 1)
                          (1.001 THRU 5.0 = 2)
                          (5.001 THRU 10.0 = 3)
                          (10.001 THRU 15.0 = 4)
                          (15.001 THRU 20.0 = 5)
                          (20.001 THRU 30.0 = 6)
                          (30.001 THRU 60.0 = 7)
                          (60.001 THRU 90.0 = 8)
                          (90.001 THRU HIGHEST = 9)(ELSE=0)
RUN SUBFILES      ALL
FREQUENCIES       INTEGER=PPUSAGE (0,9) SWAPTIME (0,9)
OPTIONS           3,6,8
STATISTICS        ALL
FINISH
*EOR
RUN NAME          CYBER THRUPUT ANALYSIS
TASK NAME         HISTOGRAM RUN OF SUBFILES
..
```

```
GET FILE         PRMFILE
VALUE LABELS     CPU (1)LESS THAN 6 SEC (2)6-12 SEC (3)12-24 SEC
                        (4)24-36 SEC (5)36-48 SEC (6)48-60 SEC
                        (7)1-5 MIN (8)5-10 MIN (9)10-20 MIN (10)MORE THAN
20/
                 TAPEIO (0)NONE (1)LESS THAN 1K (2)1-5K (3)5-10K
                        (4)10-20K (5)20-50K (6)50-100K
                        (7)100-150K (8)150-200K (9)MORE THAN 200K/

RECODE           CPU (LOWEST THRU 0.1 = 1)
                        (0.101 THRU 0.2 = 2)
                        (0.201 THRU 0.4 = 3)
                        (0.401 THRU 0.6 = 4)
                        (0.601 THRU 0.8 = 5)
                        (0.801 THRU 1.0 = 6)
                        (1.001 THRU 4.0 = 7)
                        (4.001 THRU 8.0 = 8)
                        (8.001 THRU 12.0 = 9)
                        (12.001 THRU HIGHEST = 10)(ELSE=0)/
                 TAPEIO(LOWEST THRU 0.001 = 0)
                        (0.002 THRU 1.0 = 1)
                        (1.001 THRU 5.0 = 2)
                        (5.001 THRU 10.0 = 3)
                        (10.001 THRU 20.0 = 4)
                        (20.001 THRU 50.0 = 5)
                        (50.001 THRU 100.0 = 6)
                        (100.001 THRU 150.0 = 7)
                        (150.001 THRU 200.0 = 8)
                        (200.001 THRU HIGHEST = 9)
RUN SUBFILES     ALL
FREQUENCIES      INTEGER=CPU(0,10) TAPEIO(0,9)
OPTIONS          3,6,8
STATISTICS       ALL
FINISH
*EOR
RUN NAME         CYBER THRUPUT ANALYSIS
GET FILE         PRMFILE
VALUE LABELS     THRUPUT (1)LESS THAN A MINUTE (2)1-5 MIN (3)5-10 MIN
                        (4)10-20 MIN (5)20-30 MIN (6)30-60 MIN
                        (7)60-90 MIN (8)90-120 MIN (9)2-3 HRS
                        (10)3-6 HRS (11)MORE THAN 6
RECODE           THRUPUT(LOWEST THRU 1.0 = 1)
                        (1.001 THRU 5.0 = 2)
                        (5.001 THRU 10.0 = 3)
                        (10.001 THRU 20.0 = 4)
                        (20.001 THRU 30.0 = 5)
                        (30.001 THRU 60.0 = 6)
                        (60.001 THRU 90.0 = 7)
                        (90.001 THRU 120.0 = 8)
                        (120.001 THRU 180.0 = 9)
                        (180.001 THRU 300.0 = 10)
```

```
                              (300.001 THRU HIGHEST = 11)(ELSE=0)
RUN SUBFILES     ALL
FREQUENCIES      INTEGER=THRUPUT(0,11)
OPTIONS          3,6,8
STATISTICS       ALL
FINISH
*EOR
RUN NAME         CYBER THRUPUT ANALYSIS
TASK NAME        FREQUENCY OF INTER ARRIVAL TIMES
GET FILE         PRMFILE
..
```

VITA

LeRoy Talmadge Bondurant was born in Columbus, Ohio, on Februrary 27, 1938. He graduated from high school in June, 1959, in Columbus after completing a three year tour of duty with the U.S. Army Signal Corps in Heidelberg, Germany. From 1959 to 1968 he pursued various electronic and computer technology curriculums before completing the Associate in Science, Computer Technology degree at Sinclair College, Dayton, Ohio, in September, 1970. He then graduated from Wright State University, Dayton, in December, 1972, with a Bachelor of Science in Economic Analysis. He spent the years from 1967 to 1980 as a Programmer/Analyst at Headquarters Air Force Logistics Command, Wright Patterson, AFB, Ohio before entering the Air Force Institute of Technology School of Engineering in July, 1980. He received the Master of Science in Computer Systems in December, 1981.

Permanent address: 668 Omar Circle
Yellow Springs, Ohio 45387

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/GCS/EE/81D-2	2. GOVT ACCESSION NO.: AD-A115 578	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): THRUPUT ANALYSIS OF AFLC CYBER 73 COMPUTERS		5. TYPE OF REPORT & PERIOD COVERED: MS THESIS
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): LeRoy T. Bondurant, Civ USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT/EN) Wright Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Directorate of Technical Support (AFLC/LMTA) Air Force Logistics Command Wright Patterson AFB, OH 45433		12. REPORT DATE: December 1981
		13. NUMBER OF PAGES: 172
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report): Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

15 APR 1982

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

~~FREDRIC C. LYNCH, Maj,~~ USAF Lynn E. Wolaver
Director of Public Affairs

Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Performance Evaluation
Cyber 73
Thruput Analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

AFLC's Cyber 73 Computer System is portrayed and analyzed to understand its multiprogramming and scheduling policy and the effect of those policies on the processing of the existing workload. The computer's existing workload is then characterized and represented in statistical explanatory models to show specific processing areas where job elongation could occur. The models are subsequently adjusted according to hypothesized changes to the systems hardware/software to test whether elongation points can be reduced or eliminated. Conclusions indicate a 31 percent reduction

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

in job turnaround time through the implementation of six hardware/software changes.

ATE
LMED
8